Image Cover Sheet

CLASSIFICATION	SYSTEM NUMBER
UNCLASSIFIED	511943

TITLE

Intelligent Help in the LOCATE Workspace Layout Tool

System Number:

Patron Number:

Requester:

Notes:

DSIS Use only:

Deliver to:

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
JUN 1999		**00-00-1999 to 00-00-1999**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
Intelligent Help in the LOCATE Workspace Layout Tool	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Artificial Intelligence Management and Development Cooperation,206 Keewatin Avenue,Toronto, ON M4P 1Z8 Canada,**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

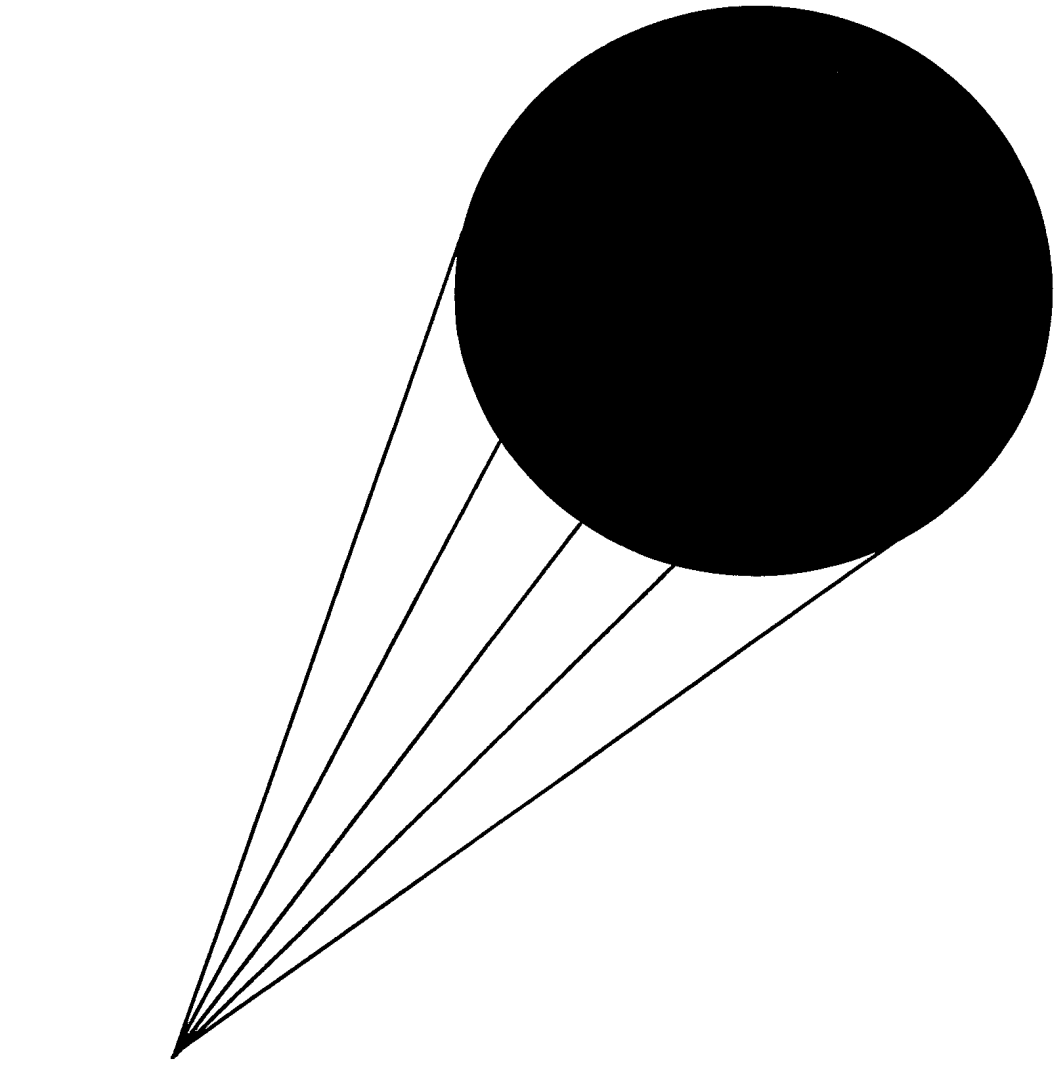
13. SUPPLEMENTARY NOTES

14. ABSTRACT

The primary purpose of this contract was to add intelligent help to the LOCATE Workspace Layout Tool. A dual approach emerged combining: 1) clever techniques that imply underlying intelligence; and, 2) a more in-depth approach that laid the groundwork for providing truly intelligent help to LOCATE users. Both approaches rely on LOCATE's ability to track user actions at the interface. The second approach identifies principal goals implied by every action taken by a user, which includes all menu. palette and object selections, all keyboard shortcuts and all text entry. Building Explicit Models was identified as a foundation for an in-depth help system for LOCATE. Consequently, a Task Model was implemented to display the results of LOCATE's action tracking and associated goal inferencing. User and System (Selt) Models were created to show LOCATE's beliefs about the user's knowledge and about its own knowledge of how it is able help users, respectively. Extensions were made to LOCATE's hypertext help system, a demonstration tutorial was created using a multimedia delivery tool and a Quick Start User Manual was developed to aid new users in working with LOCATE. Finally, as part of a proof-of-concept study for the US Navy, LOCATE was used to analyse some of their notional ICE designs. A presentation was given to US Navy staff and a report of the work was prepared under separate cover.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**Same as Report (SAR)**	**103**	

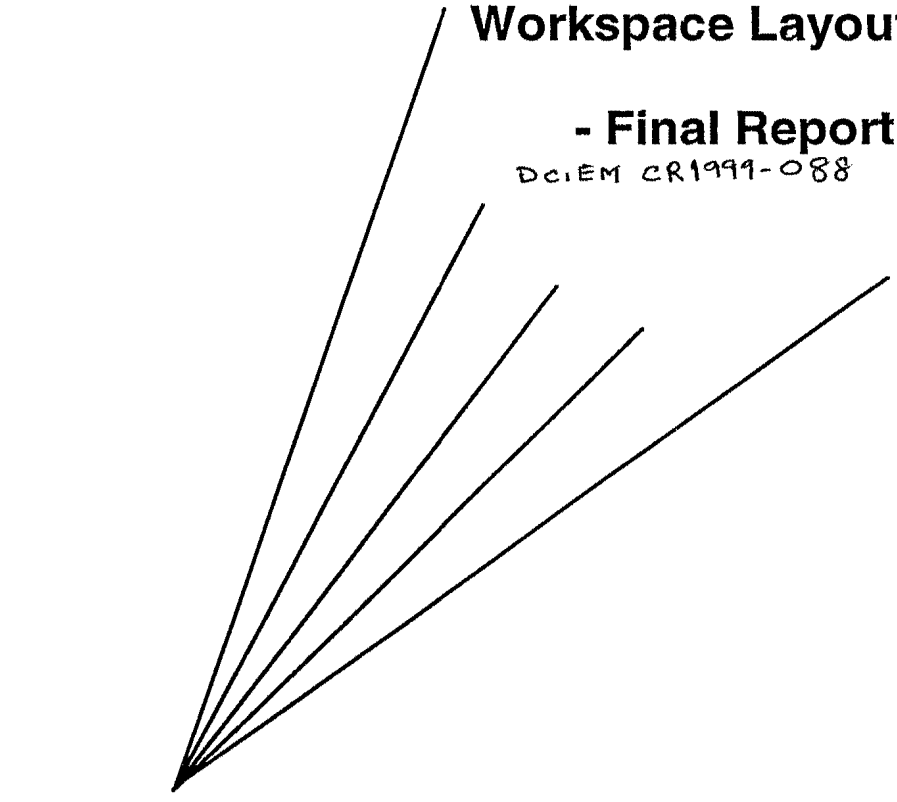


Artificial Intelligence
Management and Development Corporation

Prepared for: Public Works and Government Services
Prepared by: Artificial Intelligence Management and Development Corporation
AIM (AC196, June, 1999)
Contract: W7711-8-7480/001/SRV
"Intelligent Help in the *LOCATE* Workspace Layout Tool"
Scientific Authority:
Mr. Keith C. Hendy
Simulation and Modelling for Acquisition, Rehearsal and Training (SMART)
Defence and Civil Institute of Environmental Medicine

Intelligent Help in the *LOCATE* Workspace Layout Tool

- Final Report -



Artificial Intelligence
Management and Development Corporation
206 Keewatin Ave., Toronto, Ontario M4P 1Z8

Table of Contents

APPENDICES

FIGURES

Abstract

The primary purpose of this contract was to add intelligent help to the LOCATE Workspace Layout Tool. A dual approach emerged combining: 1) clever techniques that imply underlying intelligence; and, 2) a more in-depth approach that laid the groundwork for providing truly intelligent help to LOCATE users.

Both approaches rely on LOCATE's ability to track user actions at the interface. The second approach identifies principal goals implied by every action taken by a user, which includes all menu. palette and object selections, all keyboard shortcuts and all text entry.

Building Explicit Models was identified as a foundation for an in-depth help system for LOCATE. Consequently, a Task Model was implemented to display the results of LOCATE's action tracking and associated goal inferencing. User and System (Self) Models were created to show LOCATE's beliefs about the user's knowledge and about its own knowledge of how it is able help users, respectively.

Extensions were made to LOCATE's hypertext help system, a demonstration tutorial was created using a multimedia delivery tool and a Quick Start User Manual was developed to aid new users in working with LOCATE.

Finally, as part of a proof-of-concept study for the US Navy, LOCATE was used to analyse some of their notional ICE designs. A presentation was given to US Navy staff and a report of the work was prepared under separate cover.

Background

LOCATE is a specialised Computer-Aided Design (CAD) tool as well as a generalizable environment for creating workspace (facility) layout designs (Hendy, 1984, 1989). In addition to a graphical user interface (GUI) for workspace layout design, LOCATE is a program for evaluating the communication efficiency among design configurations. It also supports direct access to the World Wide Web and, thus, to world wide design information and opportunities for sharing ideas and solving problems among designers.

LOCATE provides workspace layout designers with the ability to analyse different types of communication within the designs they create, that is, auditory, distance (movement), tactile (reach) and visual communication. Designers create workspace elements with position and orientation and then LOCATE's default values for link functions, obstruction functions and priority weights allow them to run a cost function immediately after configuring a workspace. A cost function is a measure of communication efficiency and a user may create as many configurations as he or she wishes, compute a cost function for each and compare the results as a way of helping to determine the most efficient design.

LOCATE was originally written in DEC VAX Fortran 77, and data input was from the keyboard and very labour intensive. Subsequently, the LOCATE software was enhanced with a GUI, using Neuron Data's Elements Environment software tools. Intelligent aiding has also been added to LOCATE, supported by a rule and object-based expert system. Online help is available either locally through stored help files or globally through a demonstration help site on the World Wide Web.

LOCATE currently runs on Macintosh and PC platforms (the SGI Iris Indigo class of machine being its ultimate target platform) and generates two types of output file: those readable by LOCATE and those readable by both LOCATE and other CAD packages such as AutoCAD.

Now that LOCATE is a relatively mature tool and may be used to help solve practical problems, a next step is to provide potential users with a quick and clear way of understanding its capabilities.

As the result of work on a previous contract (W7711-6-732), users now can instruct LOCATE to track their activities and provide feedback on how they might better use LOCATE's features. Feedback to users based on such tracking is limited, however, as the work on that contract was intended as a proof-of-concept that LOCATE could be linked to the underlying expert system of Neuron Data's Intelligent Rules Element.

Other background to that work involved the development of specifications for an Intelligent LOCATE (IntL) using the theoretical approach of "Explicit Models Design" (Edwards, 1990, 1994; Edwards and Hendy, 1992). The "EM" (pronounced "M") design approach isolates four principal components of intelligent, adaptive systems, modelled as separate knowledge sources within the computer: a task model, a user model, a system-self model and a dialogue model.

In the current contract, ways of extending intelligent aiding to software applications in general and LOCATE in particular were investigated, using the concepts of system tracking of interface actions and explicit models. Some of the results of that investigation were designed and implemented into LOCATE. In addition, extensions were made to existing standard help elements of the application.

Research Approach

The general approach taken in this research involved on-going discussions with the Scientific Authority to explore ways of extending LOCATE's intelligent aiding capabilities. Discussions also centred around how best to integrate the available, standard help facilities with the enhanced intelligent aiding features. Concurrent with these discussions were the design and implementation of many of the emerging ideas.

Other topics for discussion and implementation were an on-line LOCATE tutorial and a "quick start" hard copy manual.

Study Objectives

The study objectives for this research are:

- to extend LOCATE's current hypertext help files, which provide users with an understanding of the benefits and features of the application program;
- to make use of information in the existing hypertext help system in ways that foster rapid learning among new users of LOCATE;
- to explore how intelligent aiding can be extended in LOCATE, in particular, and to other, similar applications in general;
- to extend LOCATE's current intelligent aiding capabilities by monitoring additional user interface actions and using that information to provide effective feedback to users;
- to develop basic task, user and system self models as separate knowledge sources, which will better serve the intelligent aiding process;
- to integrate some of LOCATE's existing hypertext help with its emerging intelligent aiding capabilities;
- to develop a tutorial to illustrate some of LOCATE's more important features;
- to prepare a "quick start' hard copy user manual that will serve as a brief introduction to the fundamentals of the LOCATE system;

Research Plan

LOCATE is now a relatively mature software tool with many new features added during the last development phase. Since those features have not yet been incorporated into the existing hypertext help files, a review of those files is needed to determine what should be modified and added. As that review proceeds, the hypertext help system will be upgraded accordingly.

Beyond relatively isolated descriptions of features and benefits, other types of knowledge need to be provided by the LOCATE help system. Newcomers to the software need to understand not only the types of activities supported by LOCATE but also how its various features support the performance of those activities, and, when and how they should be employed. To be a truly helpful system, LOCATE must address this variety of task information and organize it in ways that make using LOCATE for the design and analysis of workspace layouts as intuitively obvious a process as possible.

To familiarise new users with what LOCATE can do, a tutorial will be constructed to demonstrate the basics of the system. The tutorial will provide information on how LOCATE is used to design workspaces and on how it functions to help perform workspace analysis. Key features of LOCATE will be illustrated in a multimedia context.

Elements of this tutorial along with other information in LOCATE's hypertext files will be used to prepare a "quick start" hardcopy user manual.

On-going discussions will be conducted with the Scientific Authority during the course of this contract to explore ways of extending the intelligent aiding features already embedded in the LOCATE software. To assist in the development of the intelligent help system, an opportunity will be explored to apply LOCATE to the analysis of several control room layouts for the US Navy's next generation of fighting ships. During this contract, contact will be made with personnel at SPAWAR Systems Center in San Diego to more clearly establish their expectations for the LOCATE analysis (required by March 99), establish points of contact for providing the input data for LOCATE, and produce a costed proposal for the subsequent LOCATE analysis of the DD21 configurations. It is expected that one meeting with USN personnel on both the east and west coasts will be necessary.

Some of the results of discussions with the scientific authority and the USN will be translated into new and modified intelligent aiding features. Those could be as simple as the addition of an interface feature that captures summary information found useful by expert designers

when examining the results of a workspace design and analysis. Or, they could be as complex as comments and suggestions to the user that arise from monitoring his or her interface actions and reasoning about them in light of the current task and the software components for supporting that task.

Similar to work on the hypertext help system, the design and development of intelligent aiding should support the acquisition of several types of knowledge by a user, including declarative, procedural and temporal knowledge about LOCATE's many features.

Part of the current effort in intelligent aiding will involve examining the usefulness of task, user and system self models as separate knowledge sources. In a previous study dealing with how to design and build intelligent interfaces (W7711-0-7119), LOCATE served as a model application. A considerable body of specifications is available from that study that could prove a valuable resource in designing and implementing LOCATE's intelligent aiding component.

Other, already cited work on LOCATE showed how knowledge of a user's interface actions can be employed to provide feedback that can lead to a more effective use of the software. Extending that work is fundamental to understanding of how user actions relate to the task being performed and to the user's knowledge of what LOCATE can do to support that task. Designing and implementing that understanding into LOCATE will be key to providing users with reasoned, intelligent help.

Finally, discussions will be held that focus on how the principles used in building intelligent aiding into LOCATE might be generalised to other applications. Insights from those discussions should, in future, offer guidance and benefit to other, select DCIEM applications.

Adding Intelligent Help to LOCATE: A Dual Approach

Introduction

Intelligence in any entity, including a piece of software, is often a matter of perception. It is difficult, if not impossible, to know what is driving behaviour that is perceived to be intelligent, but the perception is often based on things like the novelty or complexity that is seen to characterise a set of actions. Much of the apparently intelligent behaviour of contemporary software, such as real-time correction of typing errors in a word processor, are little more than clever implementation techniques, informed by previously collected empirical data that indicate typical errors users make in typing.

As work to incorporate intelligence into the LOCATE Workspace Layout Tool proceeded, it became clear that two approaches to this general task were necessary. The first takes advantage of clever implementation techniques of the kind just described, where a designer is able to anticipate what will happen and provide for various responses based on the details of what occurs. Such "intelligent" behaviour is largely a reflection of a designer's ability to deal with contingencies involved in a user's interaction with the software.

That fact should not detract, however, from the usefulness of those techniques. They can be effective in informing users about how a piece of software works, in building their confidence in the software to help them master its features and the task at hand, and in predisposing users to want to work with the software.

The second approach to incorporating intelligence into LOCATE involves a more fundamental structuring of information as knowledge for the system. If software is able to monitor the actions taking place at the interface and represent those actions in ways that allow it to "understand" their meaning relative, say, to the overall task, then it has the potential of providing substantial and reasoned intelligent help to its users.

Once that representational groundwork has been laid, knowledge about what is happening at the interface can become part of a reasoning system for determining the kind of help to provide to a user, when that help is to be provided and the form it is to take.

Both the above approaches emerged during this contract as strategies for building intelligent aiding into the LOCATE tool and both depend heavily on LOCATE's ability to monitor user actions at the interface.

Tracking User Actions

Earlier work on LOCATE allowed for the tracking of user interface actions in two contexts. The first was in a usability study (W7711-6-7320) in which the actions of human factors experts were tracked as they gained experience with the LOCATE tool. At the end of each working session, a separate application called a, "Log Reader", summarized the results of what those experts had done while working with LOCATE, indicating primarily the type of LOCATE windows opened, how often they were opened and the duration of time they were open.

The second context in which tracking was used in LOCATE was in a proof-of-concept study to demonstrate intelligent help (W7711-6-7321). In that study, user actions were tracked and linked to rules in LOCATE's underlying expert system and those rules were then used to generate help information when certain events occurred.

The type of help LOCATE provided at that end of that study was meant only to illustrate the direction for future development. The resulting, "intelligent" help thus fell more into the category of a clever implementation technique, although it clearly made use of LOCATE's ability to monitor what the user was doing at the interface.

In the present study, the number and type of actions being monitored was substantially increased from that earlier study. LOCATE now monitors much of the user's interaction with the system including the following, which were not monitored by LOCATE in previous studies:

- all menu item selections;
- all palette item selections;
- all keyboard shortcuts;
- all selections of objects in the workspace;

LOCATE is also able to distinguish whether an object or palette item has been clicked, double-clicked or clicked in combination with some keyboard entry.

Inferring Goals from Actions

Key to understanding what is happening as a user creates and analyses a design, is a firmly based fact that most, if not all, user actions imply goals. When another person's actions give rise to inferred goals, as they do continually in human-human interaction, those inferences serve in effect to reveal the purpose behind those actions. Such inferences, when implemented in software, lay the foundation for providing truly intelligent help to the user.

A good way to describe this is with an example. Imagine that a user has clicked on a workstation in a LOCATE workspace and then has selected the "Workstation" Menu Item from the appropriate Menu (Alternatively, he could have double-clicked on the workstation object in the workspace). Either of those actions brings up a Workstation Window.

Once the window has been displayed, imagine that the user now clicks on the "Link Functions" tab at the top of that window, which displays a portion of the window that allows the user to specify link functions for the workstation.

The example below shows the goals LOCATE currently infers when the link functions tab is selected. User actions appear in black, user goals in red and system (support) goals in blue.

- Link Functions tab clicked in window of [named] workstation

U
- To change the [default link] function data for the [named] workstation

S
- To record changes in the [default link] function data for the [named] workstation

U
- To examine the [default link] function data for the [named] workstation

S
- To display the [default link] function data for the [named] workstation

U
- To show the Link Functions portion of the [named] workstation window

S
- To display the Link Functions portion of the [named] workstation window

The above goals are listed in a hierarchy that displays the goals most closely associated with the user's action at the bottom and ascends to higher-level inferred goals that provide a broader context for the action taken. Thus, the user goal, "to show the Link Functions portion of the [named] workstation window" is the last user goal listed and is supported by a LOCATE subgoal responsible for the display of that portion of the window in the interface.

The goal of examining the link function data entails the lower-level goal of displaying the link function portion of the window. Thus, the user "shows" the link function portion of the window and, having done so, immediately satisfies the next higher-level goal of examining the link function data.

Just as the goal of examining link function data required that the window on that information be opened, so the goal of changing those data depends on being able to examining them. So, once the goal of examining the data has been satisfied, the goal of changing those data may be addressed.

Of course, it may be that the user only wants to examine some of the link functions and does not intend to change them but, since LOCATE cannot be sure, it must infer both goals as likely possibilities. If the user closes the Link Functions Window without making any changes, then the goal "to change the link function data for the [named] workstation" is cancelled and recorded as such when the goal is removed from the list of current goals being pursued.

In future, probabilities could be associated with such inferred goals and subsequent actions (and goals) could be used to help confirm or disconfirm (increase or decrease probabilities) those inferences.

System Support Goals. Currently, each user goal inferred from an interface action has a corresponding system-support goal that is necessary to the realisation of the user's goal. User goals may also be subgoals or supergoals of other goals and the same may hold true for the LOCATE (system) goals. Those hierarchies of user and system goals exist but are not represented here. The only hierarchies shown are those of system goals that fall within and support a given user goal.

Other actions in the Link Functions window lead LOCATE to infer more specific goals. The example goal structure below illustrates this point. Again, supporting subgoals, adopted and pursued by LOCATE as a consequence of inferred user goals, are indicated in blue.

- Auditory-source-angular component in Link Functions (LF) window of [named] workstation clicked

U
- To change the attributes of the [named] LF component of the [named] workstation

S
- To record changes in the attributes of the [named] LF component of the [named] workstation

U
- To examine the attributes of the [named] LF component of the [named] workstation

S
- To display the attributes of the [named] LF component of the [named] workstation

- Butterworth function for the auditory-source-angular component in Link Functions (LF) window of [named] workstation selected

U
- To select the [named] function for [named] LF component of the [named] workstation

S
- To display the [named] function for the [named] link function component of the [named] workstation

S
- To display the name, "Butterworth" in the pop-up menu for the [named] link function component of the [named] workstation

S
- To record the [named] link function for the [named] link function component of the [named] workstation

As user and system support goals are satisfied, they are removed from a list of current goals being pursued. If the user clicks the Apply button instead of OK, after changing a link function or its arguments, the goals of changing that component and examining its attributes are determined to have been satisfied, and are moved into an action and goal history.

Although they are removed from the current goals list into the action and goal history, a new set of goals for revising and examining the data are instantiated and placed into the list of the current goals. This occurs because the user has left the Link Functions portion of the window open and so still may want either to examine or to make further changes to that link function component.

If the user clicks another component, however, the goal of examining the old component is tagged as satisfied but the goal of changing the attributes of that link component is cancelled.

To get a more complete idea of the extent of the goals inferred by LOCATE, Appendix A lists all of the goals currently inferred. A few last minute additions were made that are not reflected in the list in the Appendix, but otherwise the list is complete.

Using C++ to Handle Object Complexity. Much of LOCATE's functionality is derived from the original C code in which it was written, along with a rule and object-based expert system that supports its inferencing.

The creation and spawning of goals, however, is now being handled in C++. To get an idea of the nature of the abstract goal class used as a model for creating goal instances, see the goal class requirements in Appendix B. The currently implemented structure of this goal class is a subset of that shown.

Explicit Models: The Basis for LOCATE's "Understanding"

The Approach. The approach used in building a fundamental framework for intelligent help in LOCATE is that of Explicit Models (EM, pronounced "M") Design (Edwards, 1990; Edwards & Hendy, 1992; Edwards & Sinclair, in press). The main thrust of EM Design is to make explicit to the designer, the user and to the system itself, as much as is possible and useful about the nature of the task being performed, the characteristics of the human user, the characteristics of the computer and its support for the user, and the nature of the interaction between the two.

EM Design draws on aspects of several paradigms used in Artificial Intelligence and attempts to provide mechanisms for systems to model themselves through meta-abstraction and the use of metaobjects at the level of programming.

Models of Task, User and System. Following the EM Design approach, three of four general, explicit models are being designed for LOCATE: **task, user and system (self) models.** The fourth model, the **dialogue** model, may prove useful at a future date.

The task model contains information about what is happening relative to the task of creating and analysing designs. Task representation is in the form of actions and goals, specifically, the interface actions and the goals of the user and system. Example actions and goals were given in the previous section. As the user and system, action and goal history is generated and studied, it should be possible to combine it into more complex plan representations.

Further, more local representations of what the user (and system) are doing will feed rules that will decide the kind of intelligent help to provide to a user, and when and how that help should be given. Preliminary examples that point to the kind of help that is likely to emerge will be discussed presently.

User and system (self) models, like the task model, are separate knowledge sources that will ultimately inform the help system as it makes decisions about what help to give, when and where. Unlike the task model, these models are repositories of the knowledge that the user and system possess, respectively.

Two types of knowledge are the knowledge about LOCATE functionality and, less important at this stage, the knowledge about how to create and analyse designs. Of course, knowledge of LOCATE's features in many respects is closely tied to the creation and analysis of designs.

The system's knowledge about what the user knows about LOCATE's functionality is critical to providing intelligent help. Aspects of that knowledge will be covered in more detail presently.

The Object-Oriented Paradigm. The principal context for the analysis, design and implementation of explicit models is the object-oriented paradigm and a key issue in that context is the identification of the kinds of objects, including their defining attributes and behaviours, that are required to create an Intelligent LOCATE (IntL).

Key objects are already in place and are integrated with expressions of user task goals and supporting system (LOCATE) subgoals, as described in the previous section. Thus, when a user double clicks on an object in the workspace such as a workstation, LOCATE knows that the object has been selected and its attributes window opened. Further, it is able to infer a likely set of hierarchical user and associated system support goals.

Interface Issues. In the interface, knowledge contained in each model is separated into its own window. Thus, there is a Task Model Window, a User Model Window and a System (Self) Model Window.

The **Task Model Window** is divided into three sections:

- Latest Action;
- Current Goals;
- Action and Goal History.

Figure 1 below shows the form of the Task Model Window. The *Latest Action* section of that window shows the last action the user performed in the interface. The example in the window is the selection of the workstation icon in the palette.

The level of abstraction chosen for representing actions in LOCATE is that of single and double mouse clicks on identifiable objects, selections of items in menus of various kinds, keyboard commands and typed text in specified contexts.

The second section of the Task Model Window is *Current Goals* section, which provides a list of the goals currently being pursued that LOCATE has inferred from a user's action. Those goals include not only the goals the user currently holds and is pursuing but also the subgoals that LOCATE holds and is pursuing in support of the user's goals.

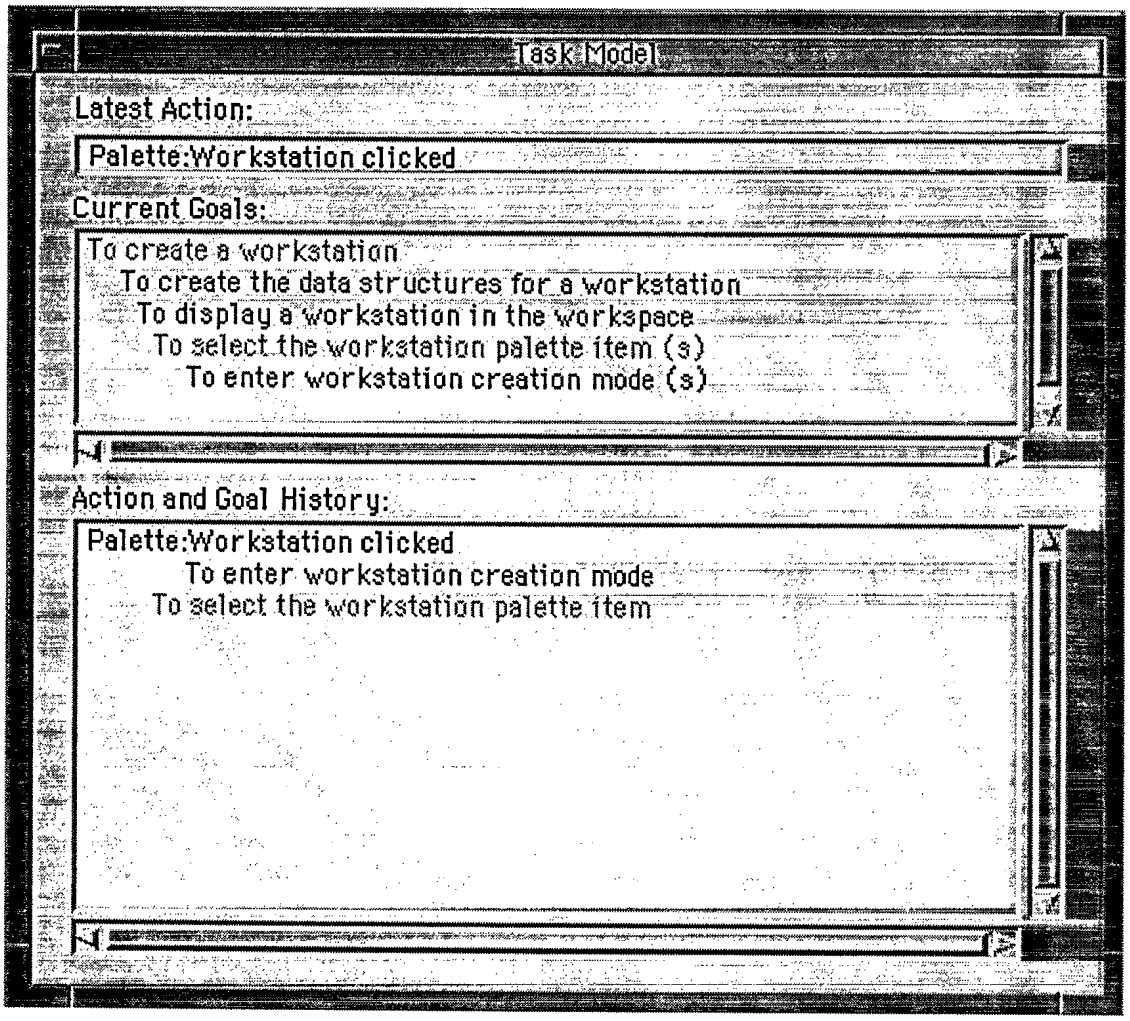


Figure 1. LOCATE's Task Model Window

Notice that the last two goals listed in that portion of the example figure have an "(s)" at the end. The "s" indicates that a goal has been satisfied. Normally, when the user performs a new action, items in the Current Goals section are transferred to the Action and Goal History portion of the window. If the goals have been satisfied, which most are, they are listed in the history with no trailing "(s)", as a way of reducing clutter. If they have not been satisfied, a "(-)" will appear after them in that part of the window. The latest action is also removed into the history area whenever a new action occurs.

The action and goal history eventually will provide an "audit trail" of everything the user and the system have done during a session. Monitoring actions and goals is a first step to a version of LOCATE that can provide truly intelligent help to its users. If the system can

"understand" what is happening, as it is happening, help to the user can be provided on that basis and can be more richly contextual.

The User Model is represented in the interface by the **User Model Window**, which contains the beliefs that LOCATE maintains about the current user.

The first information placed in that window at the beginning of each session is a question about the identity of the current user. If the user selects "About You..." from the Help Menu and fills in the information in the resulting window, LOCATE is able to answer that question.

The information requested of the user in the About You..." window is shown in Figure 2.

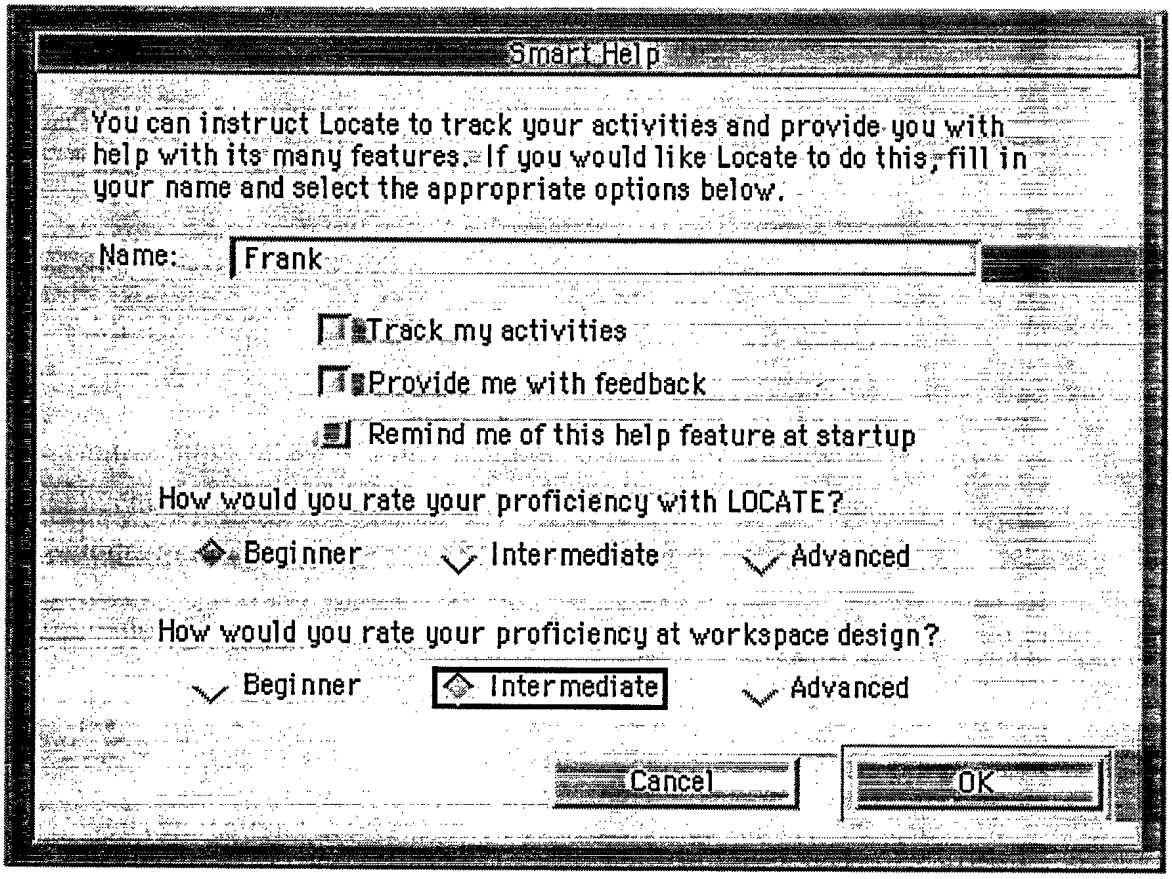


Figure 2. LOCATE's "About You..." Window

Having now been informed about the identity of the user, LOCATE is able to answer its initial question about the user's identity. LOCATE's initial question and the resulting answer are shown in LOCATE's User Model Window , an example of which appears in Figure 3.

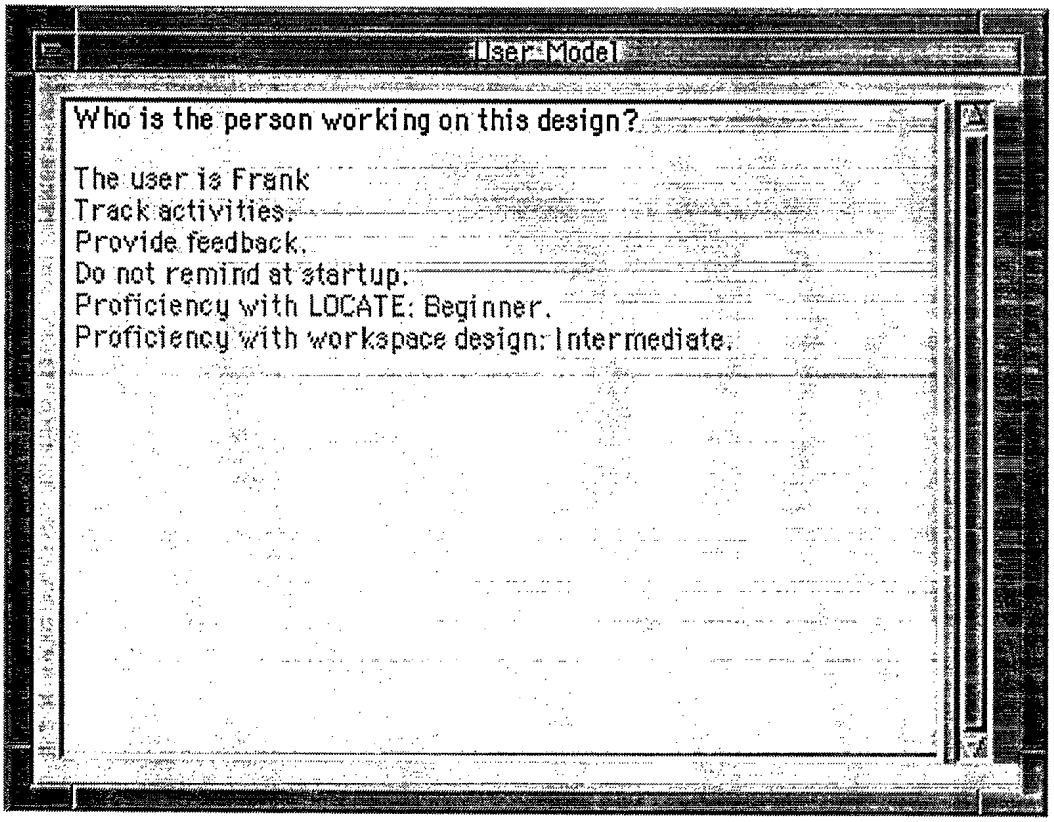


Figure 3. Example Question and Answers Regarding a User's Identity in LOCATE's User Model Window

The information in above example of the User Model Window represents only one source of knowledge that LOCATE uses to acquire information about what the user knows, namely, knowledge derived from what it has been told by the user.

This along with the other two sources of LOCATE's knowledge about the user are discussed more thoroughly in the next section.

Providing Intelligent Help

As mentioned earlier, the system's knowledge about what the user knows about LOCATE's functionality is critical to intelligent aiding. That knowledge is gained in three ways by LOCATE:

- direct statements by a user about what he or she knows (see Figure 3 for an example);
- observation of user actions and the consequent inferences that can be drawn;
- help information provided to the user by LOCATE.

Knowledge Derived from Direct User Input. Currently, some minimal information is obtained directly from the user. If the user decides to have LOCATE track his activities, he needs to identify himself to the system through his name, initials, etc.. At that time, the user can ask LOCATE to track his activities and to provide feedback. Further, he can ask that LOCATE remind him of this identification and tracking feature each time LOCATE is opened. All of this information is obtained directly from the user when he selects "About You..." from the Help Menu (see Figure 2 above, for an example of that window).

Knowledge Based on System Inferences from User Actions. With respect to observation and inference, LOCATE is able to track a few user activities and provide some clever help. One example from an earlier study is LOCATE's ability to know whether the user is opening windows by double-clicking or using a menu item selection. If the latter, then, after several instances of menu selections, LOCATE puts up a help window indicating to the user that a specific window or, more generally, object windows, can be opened by simply double-clicking on the object. Figure 4 below shows such an example help window.

Knowledge Deduced from Help Provided to the User by LOCATE. Knowledge about the user is derived not only from such observations but also from the results of LOCATE's presentation of help information.

In the above example, after putting up the help window, LOCATE can conclude that the user likely knows how to open workstation windows using the double-click method.

Displaying this window and obtaining the user's answer as to whether he or she knew that double-clicking on a workstation would open its window means that LOCATE can now draw inferences about what the user likely knows and can store that knowledge in its User Model.

An example of such knowledge appears in Figure 5, with LOCATE's comments and inferences in black, LOCATE help in blue and user responses in red.

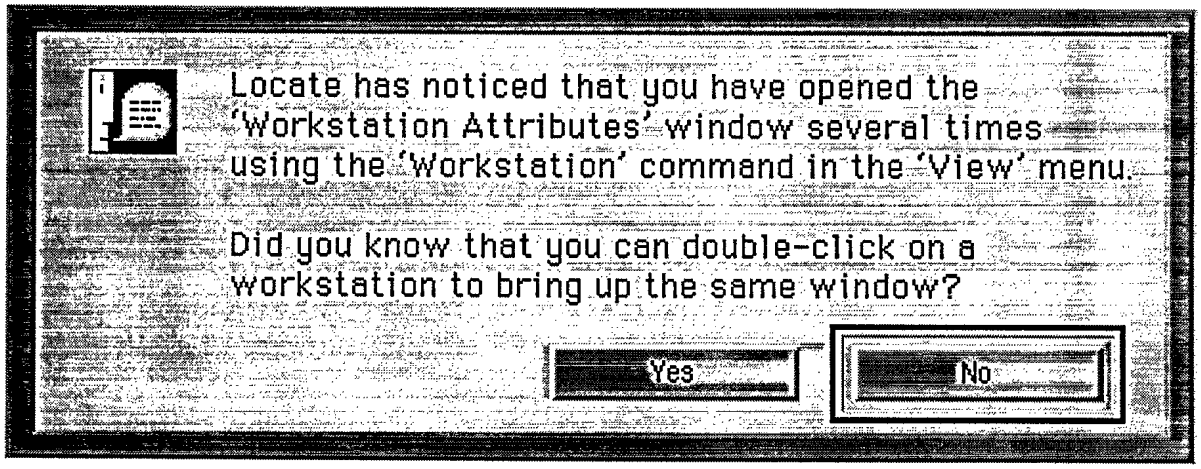


Figure 4. Help Window Displayed as a Result of Monitoring User Interface Actions

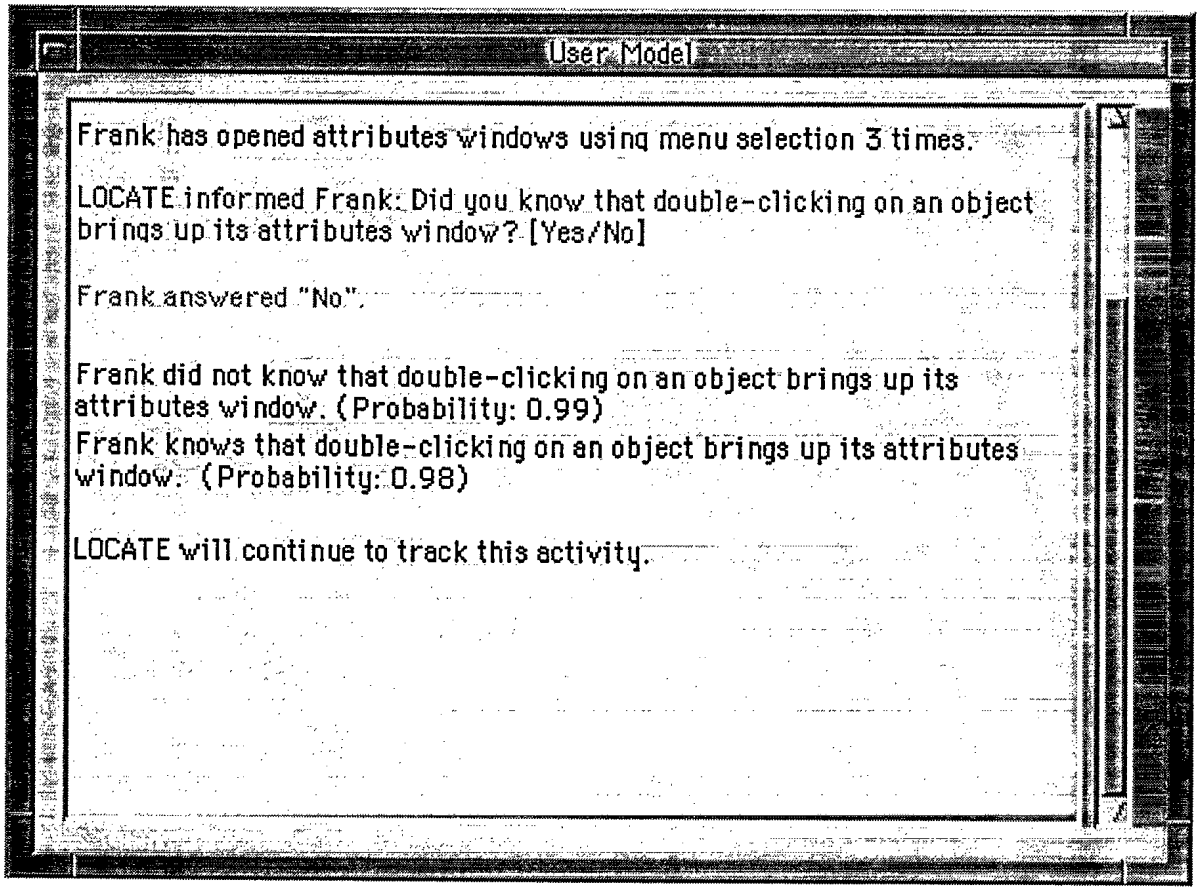


Figure 5. Example Information Placed in LOCATE's User Model Following a User's Answer to the Question in the Help Window Shown in Figure 4

Detecting the subsequent use of menu item selection will now have different implications for help than it did before. If the user continues to use menu selection to open workstation and other windows, LOCATE will be led to the conclusion that he or she may prefer that method over the more efficient, double-clicking.

In addition to knowledge about the user, intelligent help and analysis can arise from information stored in a system (self) model.

The knowledge a system can have about itself includes knowledge (beliefs) about its own capabilities. The primary reason for creating such a model is to aid the user and so, the knowledge contained in a system (self) model should be organised around providing intelligent help and analysis to users about the features of the LOCATE system.

For that reason, early work on LOCATE's system model links a new Smart Help Menu item with its System Model. Further, it was determined that much of the help that LOCATE might offer to a user could take advantage of LOCATE's goal orientation as represented in its Task Model.

An example of that kind of help appears in Figure 6 on the next page. The window in that Figure is displayed whenever the user selects "Smart Help" from the Help Menu. Smart Help is organised around a concept of "How to...."

In the example, the user has asked LOCATE how to print and LOCATE responds to that query by displaying a list of goals associated with the concepts of "print" that it understands. That list is displayed in a panel entitled, "Goals" on the left side of the window.

The list of goals provides the user with choices that focus his inquiry. When the user chooses one of the options in the list, a set of subgoals (a rudimentary plan) appears on the right which tells the user how to achieve the goal on the left..

In the current example, the user has selected "To print a design" and the system has displayed a list of subgoals on the right that show the user the subgoals *qua* actions that are necessary for him to be able to print a design.

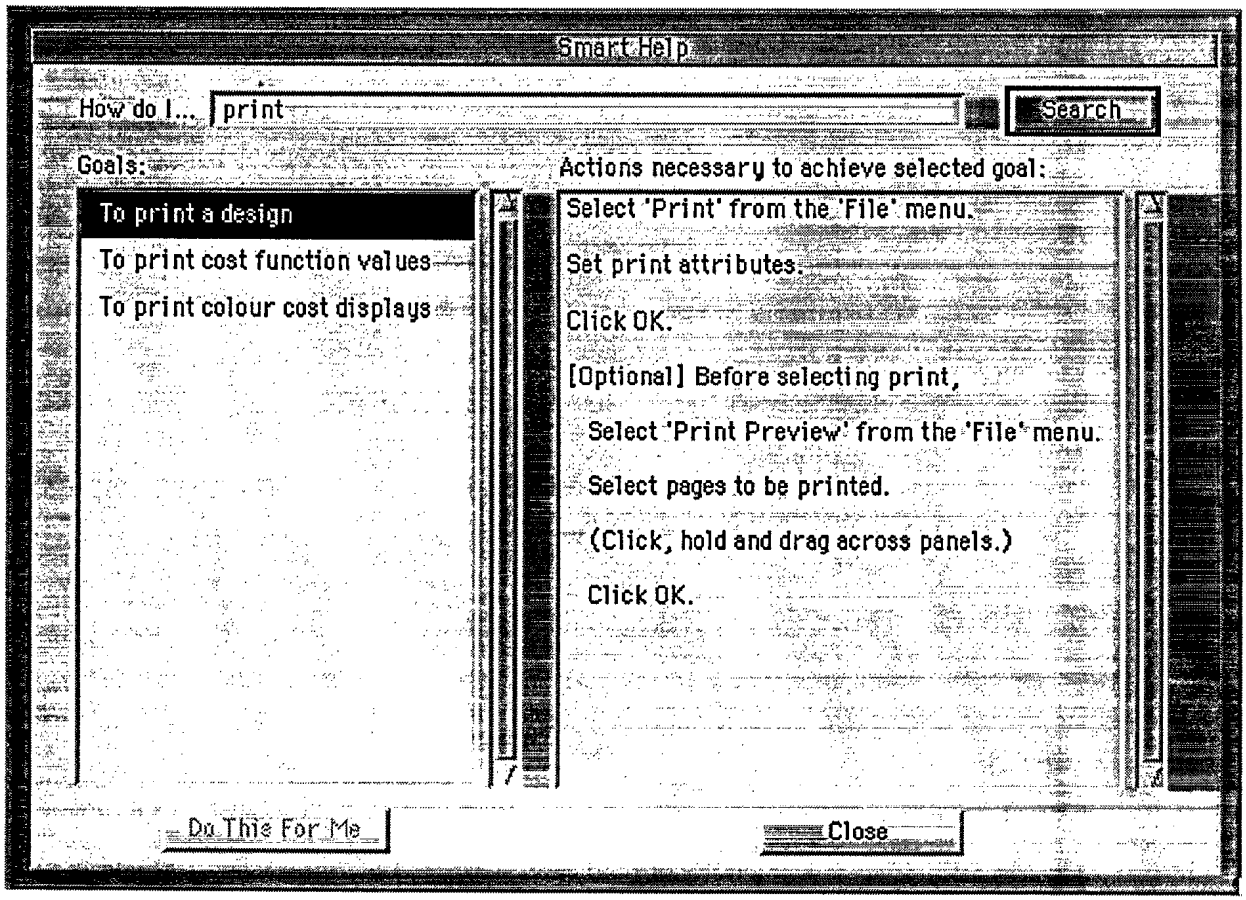


Figure 6. Example of LOCATE's Smart Help Window

Notice that the query from the user is in terms of an action, the high-level response from LOCATE in terms of goals and the consequent plan framed in a panel entitled, "Actions necessary to achieve the selected goal." Of course, those actions are a mixture of subgoals, conditionals and "primitive" actions among other things, but that mixture is constructed by LOCATE in a way that makes sense to the user.

Just as the task model reflects an understanding of what is happening in the interface through goal and plan recognition, LOCATE's help and analysis facilities support the user through a process of goal and plan generation. For recent work elaborating intelligent help as plan generation, see Küpper and Kobsa (1999).

As the user requests and receives help from LOCATE, the system's system (self) model tracks what is taking place. Figure 7 shows the consequences for the system model of the interaction that has just occurred in the Smart Help Window.

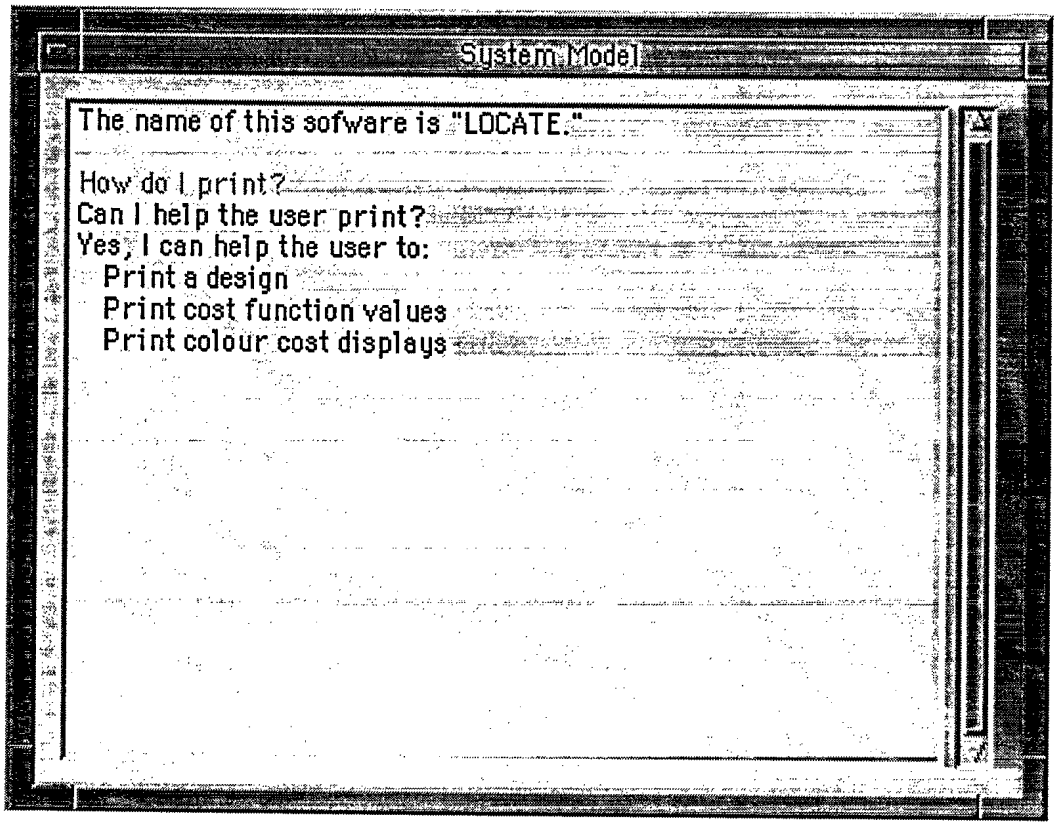


Figure 7. Consequence for System (Self) Model of User Query in Figure 9.

The first entry in the window is a standard entry placed there at startup, which simply identifies the system (in its own self model) for the user. "The name of this software is LOCATE." A first party view was thought to be too anthropomorphic.

The next set of entries reflects the user's request for help. These entries are constructed in a way that is meant to reflect a sense of self enquiry on the part of the system, relative to whether it is able to help the user with some identified problem.

This approach has several advantages. First, it provides an intuitive way for a developer to track the system's exploration of its own knowledge and capabilities relative to user enquires. It also permits the developer to request greater detail about that exploration at any point in the process.

For a user who might wish to view the system model, it communicates that the central purpose of self enquiry on the part of the system is to help the user better achieve his or her

goals. It also may be that both the content and nature of such a self-enquiry approach is non-threatening to users in the context of a system that keeps track of its own knowledge and capabilities.

In summary, information about what the user knows is stored in LOCATE's User Model, based on the three sources of knowledge identified above: direct information from the user, observation of user interface actions and knowledge about what help information LOCATE has previously provided. System Model information is organized around what the system knows about itself that can help users answer their questions and solve their problems.

Hypertext Help

In a previous study (W7711-6-7321), a hypertext help system for LOCATE was designed based on a set of over two hundred information categories. The system was created using the PageMill Web authoring tool from Adobe. Such tools speed coding of text that has been prepared in word processors such as Microsoft Word and make the integration of graphics a relatively easy task.

The coding conformed to HTML 2.0 standards, which means that LOCATE's help system complied with the then current hypertext requirements for display on the World Wide Web.

Building the hypertext system for LOCATE required decisions regarding how information was to be categorised, organized, distributed and related within a hypermedia environment. The organizational chart on the next page (Figure 8) provides an overview of those decisions and conforms largely to the information categories just mentioned.

During the current contract, revisions were made to the approximately forty hypertext help "pages" on LOCATE. Because of the nature of the material, hardcopy printouts are not provided for either the page displays or the associated HTML code. The entire hypertext help system and documentation may be displayed by accessing the help files locally or by logging in to the Internet and browsing the Web site at the following location:

http://www.interlog.com/~jle

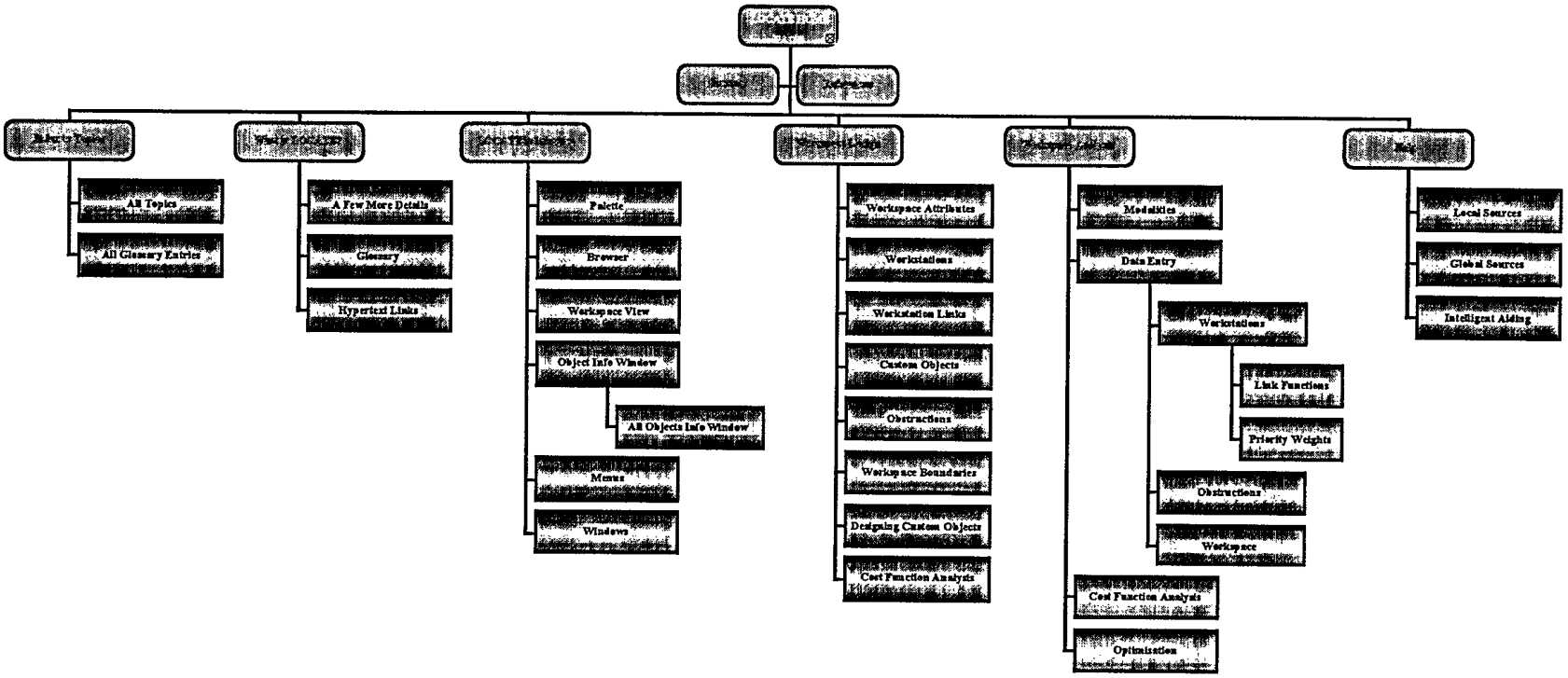


Figure 8. Overview of LOCATE's Hypertext Help System

Two versions of the LOCATE help system are currently supported. The first version makes use of background colours, frames and tables, HTML options not currently supported by the Web Browser that is integrated with the LOCATE application. The former is the one stored at the above Internet Web site.

A second version that has been scaled down and that will display in LOCATE's own Browser has been included as a separate set of files that can be stored locally. Originally, in order to take advantage of the Browser, the files had to be located in the same directory as the Elements Environment library files. Work on this contract now permits LOCATE to access the files stored in a "Help folder."

Unfortunately, Neuron Data has chosen not to expand the functionality of its build-in Browser and has incorporated (read, "hidden") it within its interface component. This is unfortunate, if understandable, given the dominance of Netscape and Microsoft's Internet Explorer among competing Browsers. What this means is that any code extending the functionality of Neuron Data's Browser must be customised code.

It is recommended, before any further work is conducted to extend the existing Neuron Data Browser, that options be explored for how LOCATE might be interfaced with the two dominant Browsers on the market. Given the possible development of "plug-in's" for the two Browsers, LOCATE could access fully functional Browsers and considerable time, effort and money would saved by avoiding the need to develop and customise the Neuron Data Browser.

Conversion of the LOCATE Thesis. As a result of cost savings on this contract, approval was obtained to use the windfall to convert Hendy's (1984) thesis, which is the foundational work of LOCATE, to electronic form.

Two word processing firms were contacted and one was found that was agreeable to a fixed price for the work of typing the thesis. The decision to accept a fixed price was fortunate since the typing of the thesis took much longer than had been anticipated. This was largely due to problems arising from the use of Microsoft Word on a PC platform; the application on the PC has less flexibility in generating equations than the Macintosh and many equations had to be created more than once. The final proofing of the thesis was done after the entire document had been transferred to the Macintosh and numerous formulas edited and revised.

It had been hoped that once the thesis was typed that it could be converted to HTML and incorporated as part of LOCATE's hypertext help system. Due to the longer-than-expected time to complete the typing of thesis, this was not possible.

An attempt was made, however, to export a few pages of the document using Microsoft Word's HTML conversion facility. Word's conversion involves creating an HTML document for standard text and then converting each equation into a separate image file. In trying to do this with only a few pages, Word encountered memory problems and was not able to proceed.

In future, it may be possible to convert the text to an Adobe Acrobat format for access over the Internet as an entire document but, currently, it is not clear how easily the equations and graphics will port to that application. Also, by making it available as an entire document, relevant portions cannot be easily accessed, if at all, through hypertext links from other Internet Web pages.

At some future date, no doubt, it will be useful to separate the thesis, page-by-page, and then try and use Word's conversion routine to create HTML representations for each page and image files of its equations. That will give the kind of referencing flexibility needed if the thesis is to be integrated with LOCATE's intelligent aiding capabilities.

Demonstration Tutorial

To familiarise new users with what LOCATE can do, a tutorial was planned to demonstrate the basics of the system. The tutorial was to provide information on how LOCATE can be used to design workspaces and on how it functions to help perform workspace analysis. Key features of LOCATE were to be illustrated in a multimedia context.

The software selected for implementing the tutorial was Macromedia's *Director*, which provides a means of constructing full-featured hypermedia supporting bitmaps, vector shapes, text, scripts, animation (frame-by-frame or tweening), sound, Flash movies, QuickTime, digital video and AVI video.

In addition, Macromedia Director supports a full-featured programming language called, "Lingo", which is much like Apple's HyperTalk used with its Hypercard application, except that Lingo has greater flexibility.

After acquiring and studying Director it became clear that it would be possible to construct only a limited tutorial as a first pass. In fact, the tutorial that has emerged is more of a demonstration to familiarise users with the benefits and features of the LOCATE software.

The content of that demonstration tutorial is a converted Powerpoint file used in a recent LOCATE presentation. Director provides two key capabilities which permitted that presentation to be used. First, it allows conversion of Powerpoint files to Director files and, second, it allows those files to be saved as (hypermedia) HTML files, which then can be uploaded the Internet .

The conversion of the Powerpoint file was only partially successful due to the requirement of saving the file in an older version and considerable work was required to make the file as close as possible in appearance to the original, to upload it to the Internet and to have it run successfully.

The tutorial may be viewed visiting the Internet demonstration site mentioned earlier, which contains the LOCATE hypertext help files:

http://www.interlog.com/~jle

Quick Start Manual

Elements of the tutorial along with other information in LOCATE's hypertext files were used to prepare a "quick start" hardcopy user manual. The manual is a brief introduction to LOCATE that can be delivered to users along with the application. It describes key benefits and features and helps users quickly apply LOCATE to their design problems.

The quick start manual exists as a separate document and also appears in Appendix C.

LOCATE Analysis of US Navy ICE Designs

As part of the current contract, a proof-of-concept study was performed on a set of ICE (Integrated Command Environment) designs made available to AIM by the US Navy. The ICE work is part of the US Navy's SC-21 (Surface Combatant- 21) Project, which has as its focus the design of the Navy's warships for the 21st century.

The study involved populating LOCATE workspaces using downloaded DXF files for three ICE design configurations:

- Amphitheatre;
- Arena;
- Table Top.

Cooperative work with a US Navy Subject Matter Expert (SME) provided additional information necessary to performing LOCATE analyses. Those analyses served to compare the efficiency of the three ICE designs across the four LOCATE-supported communication domains: vision, audition; touch (or reach) and distance (or movement).

Because the data used in the study were subject to a variety of limitations, the numerical results were considered suggestive at best. In contrast, the **most important findings** of the study were:

- that LOCATE was able to generate metrics that provide for quantitative comparisons of the communication efficiency of the US Navy's notional ICE designs;
- that clear conclusions can be drawn about the relative efficiency of those designs within the context of assumptions made about the players involved and their communication patterns;
- that features of LOCATE, such as its colour cost displays for quickly identifying, high, medium and low pairwise costs between workstations (players), support effective decision making about how designs might be modified to reduce communication costs.

Keeping in mind the fact that there were a number of limiting factors on the data and that the important findings emerged from the process of applying the tool, results of LOCATE analyses showed the Table Top to be the most efficient configuration, followed by the Amphitheatre and then the Arena.

The ability to draw such conclusions in a timely manner provides a context in which ideas often quickly emerge as to how to make the "best" configuration even better (more efficient). By way of example, after running the analyses, it was possible to modify the Table Top configuration in this study to produce a more efficient configuration.

Some of the limitations on the data mentioned above include the following:

- the designs were notional;

- only three ICE designs were compared;
- the three designs chosen, initially contained different numbers of players. In order to compare configurations, LOCATE assumes a common set of players for all configurations. After consultation with the SME, eight players were identified as common to each design. Standardising the configurations involved adding, moving and combining players;
- variations in workspace size also occurred among configurations. Similar to the number of players in the configurations, LOCATE assumes workspace size to be the same for all designs. Consequently, each ICE design was assigned a workspace size equal to that of the largest configuration;
- measurements within designs, such as positioning and angle of rotation of consoles, were not directly available and had to be extrapolated from object characteristics in the imported DXF files;
- key data were of a preliminary nature and were based on assumptions about communication patterns among the players in the three configurations;
- wall and table displays were not represented in the analysis;
- no object was identified as an obstacle to communication, although some items, such as the table in the Table Top configuration, might qualify.

Instructive notes on some of the assumptions underlying the choice of link function and priority weight data used in the ICE design study appear in the "Assumptions..." sections of Appendices C and D of the report. That report, submitted to DCIEM under separate cover is entitled, *Applying the LOCATE Workspace Layout Tool to the US Navy's Notional ICE Designs.*

More detailed knowledge about the characteristics of the designs and the patterns of communication among players would certainly lead to changes in the data used to analyse communication efficiency.

The results of this study were presented to representatives of the US Navy in a meeting at DCIEM in March, 1999. As a result of that presentation, an invitation was initiated by the

Navy to both DCIEM and AIM to participate in a one-week workshop in which LOCATE would be used to help analyse a new set of ICE designs.

Modifying LOCATE based on the US Navy Work. In addition to illustrating how LOCATE can be applied successfully to a practical problem, the ICE Design study revealed several new features that were incorporated into LOCATE to support the importing of DXF design files.

A key feature now allows users to specify aspects of a design just prior to importing. The window in Figure 9 below is displayed whenever the user chooses "Import DXF file" from the File Menu and then selects a DXF file to import.

On the left side of the window are default X and Y dimensions and a scaling factor for the DXF file to be imported. LOCATE determines the dimensions by examining the contents of the DXF file and computing the largest bounding rectangle that will encompass the objects in the design. The scaling factor is an arbitrary value and probably should default to "1."

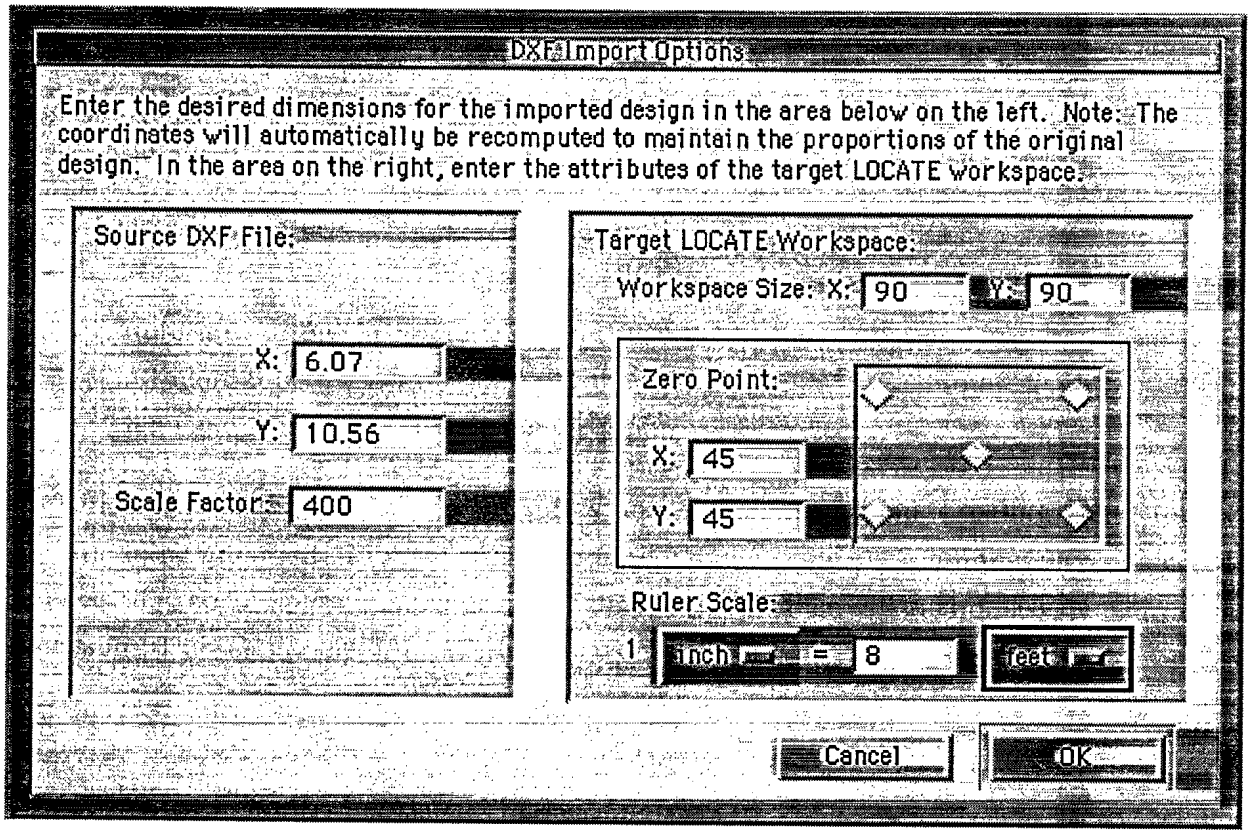


Figure 9. Example DXF Import Options Window.

The dimensions and the scaling factor are not independent of each other. In the case of the Navy designs, setting the scaling factor to "1" changes the X and Y dimensions to approximately 2400 and 4200, respectively. By setting the scaling factor to "1", then, a user can see just how large is the design to be imported and, thus, better determine the scaling factor and dimensions he needs to use for importing that design.

On the right side of the window, the user can set the measurement unit that LOCATE is to use, the scaling ratio, boundary dimensions and the zero point of LOCATE's workspace.

As indicated, the example figure above shows the dimensions and scaling factor used in importing the Navy's ICE Designs. The information on the right side of the example window shows the subsequently set LOCATE workspace size, zero point and, finally, the unit of measurement and ratio scale, which was determined to be one inch = eight feet for the imported Navy designs.

When importing designs, the user may have to experiment and import them more than once in order to get a "fit" with which he is satisfied. By providing feedback on the dimensions of a design prior to importing it and allowing the user to coordinate that with the LOCATE workspace dimensions, the job of getting the right "fit" is made considerably easier.

Summary

Work on this project had considerable scope, from incorporating much of the groundwork necessary to intelligent aiding to conducting a study in which LOCATE was applied to some notional designs provided by the US Navy.

The major accomplishments of this work are as follows:

- the extension of LOCATE's current hypertext help files;
- the exploration of how intelligent aiding can be extended in LOCATE;
- the expansion of the type and number of user interface actions that LOCATE is able to track;
- the identification of goals likely implied by each of the actions a user takes at the interface;
- the representation, organization and presentation of those actions and goals in a Task Model Window, thus providing LOCATE with an understanding of what the user and LOCATE are doing in the process of design and analysis, when they are doing it and how;
- the representation, organization and presentation of "user knowledge" in a User Model Window, providing LOCATE with a view of the user's current understanding of LOCATE's functionality;
- the representation, organization and presentation of system (self) knowledge in a System (Self) Model Window, providing LOCATE with a deeper understanding of its own capabilities;
- the identification and elaboration of a dual approach to intelligent aiding in LOCATE. That approach includes basing help both on an in-depth understanding of the task at hand and on clever techniques to aid the user;
- the development of a LOCATE demonstration tutorial, implemented in Macromedia's multimedia application environment, *Director*;
- the preparation of a quick start" hardcopy user manual;

- the creation of an electronic version of the original thesis on which LOCATE is based;
- the application of LOCATE to some notional designs provided by the US Navy as part of its Integrated Command Environment (ICE), which forms part of the Navy's SC-21 (Surface Combatant-21) project. That project investigates the design and development of the US Navy's warships for the 21st century.

Although the principal goals of this project were achieved, a few objectives met with only partial success. Those include:

- making use of information in the existing hypertext help system in ways that foster rapid learning among new users of LOCATE;
- integrating some of LOCATE's existing hypertext help with its emerging intelligent aiding capabilities;
- the construction of a demonstration tutorial;
- the inability of integrating the Hendy thesis into LOCATE's hypertext help system.

The first two were limited because the focus of the current work revolved largely around laying the groundwork for truly intelligent help through implementing goal inference from user actions. The third objective was limited by the unanticipated complexity of newly acquired Director software and time constraints. A final limitation relating to the desire to integrate some or all of the Hendy thesis into LOCATE hypertext help system occurred because of unanticipated delays caused by Microsoft Word's ability to handle equations on a PC platform.

In spite of these limitations, the current efforts have proved successful in clarifying key notions about how intelligent aiding can proceed in LOCATE and in designing and implementing the groundwork necessary for that to happen. Although some additional work remains to be done on the groundwork, i.e., identifying goals for specific actions in all of LOCATE's windows, the next major thrust can focus on how that groundwork can be used effectively to aid LOCATE's users.

Future Work

The next steps in the design and development of LOCATE include:

- continuing to develop and refine LOCATE's basic design and analysis features;
- commercialising the application;
- expanding the groundwork for tracking actions and goals at the interface;
- designing and implementing rules in a way that takes advantage of LOCATE's new "understanding" of what tasks are being performed, when and how, and that uses that information to provide truly intelligent help to the user,
- expanding LOCATE's understanding of the user's knowledge of LOCATE's functionality and of workspace design and analysis;
- extending its knowledge of its own features so as to better help users in a variety of design and analysis contexts.

These steps could be accomplished in two different ways. First, by taking a broader approach to intelligent aiding that would involve generalising LOCATE's principles of intelligent aiding to other development efforts in DCIEM's lab, and beyond. Such an approach could be realised through an intelligence testbed that would serve as a context in which design and implementation techniques for intelligent aiding would of explored for several application development efforts, including LOCATE.

Practical results from the testbed could be incorporated into the various applications and that work could lead to further generalisations for how to modularise intelligent help for easier delivery to a variety of applications.

Second, more focused work is still needed on LOCATE's basic functionality. Although LOCATE is a relatively mature tool that is beginning to prove itself in real-world applications, there are still a number of features that are needed to "round out" its basic functionality and there are still a few problem areas that remain to be addressed.

Regarding the latter, more exhaustive tests are needed on the new manual grouping feature and its interaction with other components in the LOCATE system. As to the former, items are included like vertical and horizontal object alignments, a snap-to grid, an expanded tool library, more work on issues related to importing and exporting DXF files, and a resolution as to how best to link LOCATE to the World Wide Web and, thus, to world-wide design information in a way that will be productive for users conducting LOCATE analyses.

While these features are important in rounding out LOCATE as a mature design and analysis tool, it may now be possible to consider shifting the central focus away from the development of the tool itself toward using LOCATE as the basis for building a testbed for intelligent aiding that will be useful to a wide range of application development efforts.

References

Bass, E.J., Ernst-Fortin, S.T., and Small, R.J. (1997). Knowledge base development tool requirements for an intelligent monitoring aid. *Proceedings of the Tenth Annual Florida Artificial Research Symposium (FLAIRS-97)*, Daytona Beach, FL, May 12-14, pp. 412-416.

Broadbent, G. (1988). *Design in architecture.* London: David Fulton Publishers.

Edwards, J. L. (1990). Intelligent dialogue in air traffic control systems. In J. A. Wise, V. D. Hopkin and M. L. Smith (Eds.), *Automation and systems issues in Air Traffic Control.* New York: Springer-Verlag.

Edwards, J. L., & Hendy, K. (1992). Development and validation of user models in an air traffic control simulation. Paper presented at the Second International Workshop on User Modeling. International Conference and Research Center for Computer Science (IBFI), Dagstuhl Castle, Germany, August 10-13, 1992.

Edwards, J. L., & D. W. Sinclair (1999, in press). Designing intelligence: A case of explicit models and layered protocols. In M. M. Taylor, F. Néel and D. Bouwhuis (Eds.), The Structure of Multimodal Dialogue II. North-Holland.

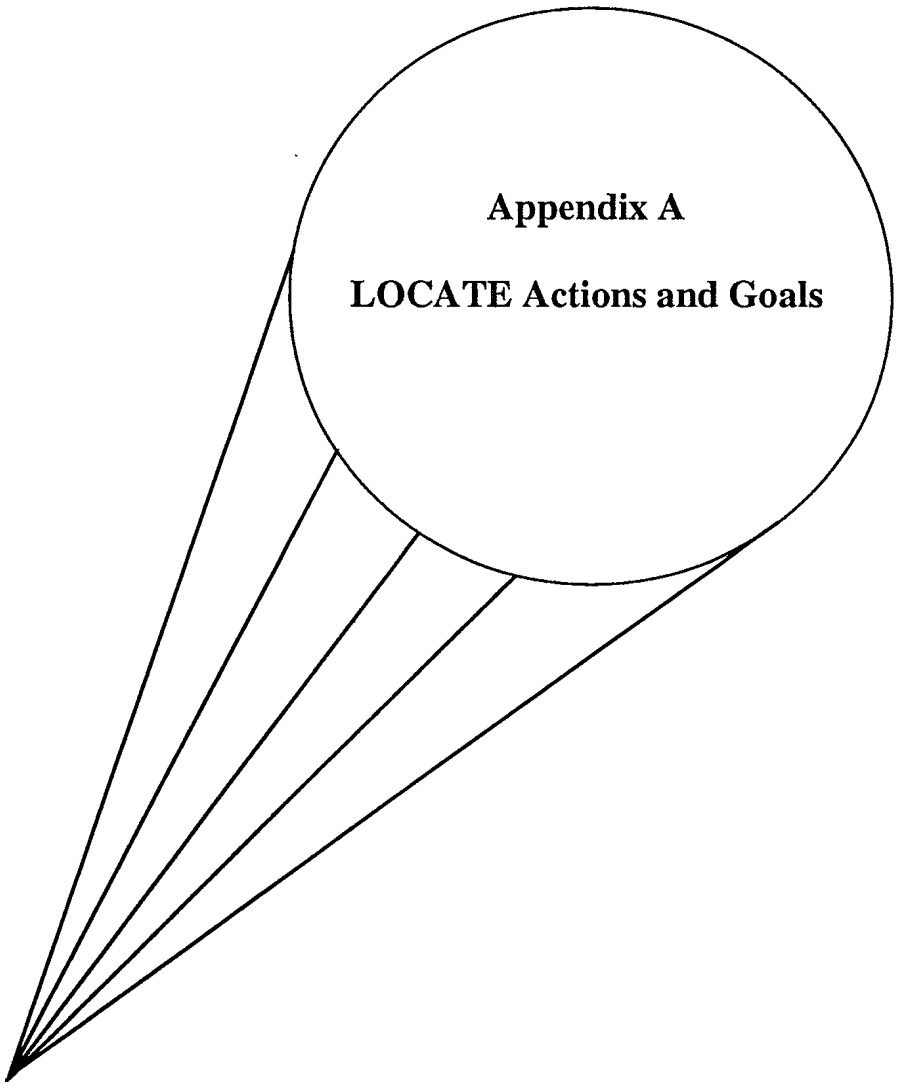
Hendy, K. C. (1984). 'Locate': A program for computer-aided workspace layout. Master's Thesis, Department of Electrical Engineering, Monash University, Clayton, Victoria, Australia.

Hendy, K. C. (1989). A Model for Human-Machine-Human Interaction in Workspace Layout Problems. *Human Factors, 31(5),* 593-610.

Polson, M.C., & Richardson, J.J. (Eds.) (1988). *Foundations of intelligent tutoring systems.* Hillsdale, N.J.: Lawrence Erlbaum.

Spector, J.M., Polson, M.C., and Muraida, D.J. (Eds.) (1993). *Automating instructional design: Concepts and issues.* Englewood Cliffs, N.J.: Educational Technology Publications.

VanLehn, K. (1988). Student Modeling. In M. C. Polson & J. J. Richardson (Eds.), *Foundations of intelligent tutoring systems* (pp. 55-78). Hillsdale, NJ: Lawrence Erlbaum Associates, Inc.



Appendix A

LOCATE Actions and Goals

Artificial Intelligence
Management and Development Corporation

The following is a list of actions and goals LOCATE is now capable of tracking. The list is preliminary and will be revised over time. It provides a foundation for understanding what the user and system are doing with respect to the task of design and analysis. Its ultimate goal is to serve as a basis for providing users with intelligent help from an in-depth understanding of what the user is doing, when and how.

A variety of low level goals are not included, e. g., selecting menu items, moving the mouse to a particular location, clicking or double-clicking (on an object) in the workspace, on a palette item or in the Cost Function Value window, selecting check boxes or radio buttons, typing information in specific edit text boxes within specific windows, etc.

The actions and goals at that level will be included in future work on LOCATE. At that level of abstraction, System support goals are easily seen as "feedback." Feedback is a much broader concept, however, and likely exists at most levels of User and System goals.

- **LOCATE Actions and Goals by Menus**

  File Menu

  - New
  - (Ctrl-N)

U • To create a new design

S • To display an empty Workspace

U • To specify the attributes of the new workspace

S • To record attributes of the new workspace

U • To open the Workspace Attributes window

S • To display the Workspace Attributes window

  - Open
  - (Ctrl-O)

U • To open an existing design

S • To input the file from disk

U • To open the Standard Open dialox

S • To display the Standard Open dialox

  - Save
  - (Ctrl-S)

U • To save a design

S • To output the file to disk

U • To open the Standard Save dialox (if untitled)

S • To display the Standard Save dialox (if untitled)

  - Save As...

U • To save a design

S • To output the file to disk

U • To open the Standard Save dialox

S • To display the Standard Save dialox

- Page Setup

U
- To set page attributes

S
- To alter page attribute data structures

U
- To open the Page Setup dialox

S
- To display the Page Setup dialox

- Print Preview

U
- To set printout attributes

S
- To alter printout attribute data structures

U
- To open the Print Preview window

S
- To display the Print Preview window

- Print
- (Ctrl-P)

U
- To print a document

S
- To output the document to the printer

U
- To set [other] printout attributes [specific attributes to be tracked...eventually]

S
- To record printout attribute

U
- To open the Print dialox

S
- To display the Print dialox

- Import DXF...

U
- To import a DXF design

S
- To input and translate the file from disk

U
- To open the Standard Open dialox

S
- To display the Standard Open dialox

- Export DXF...

U
- To export a DXF design

S
- To translate and output the file to disk

U
- To open the Standard Save dialox

S
- To display the Standard Save dialox

- Quit
- (Ctrl-Q)

U
- To quit LOCATE

S
- To exit from the application

Edit Menu

- Undo
- (Ctrl-Z)

U
- To undo (or redo) an action

S
- To update the display of the object

S
- To store data in the undo buffer (or retrieve data from the undo buffer)

- Cut
- (Ctrl-X)

U
- To cut an object

S
- To undisplay an object

S
- To delete an object's data structures

S
- To store an object to the clipboard

- Copy
- (Ctrl-C)

U
- To copy an object

S
- To store an object to the clipboard

- Paste
- (Ctrl-V)

U
- To paste an object

S
- To display an object

S
- To recall an object from the clipboard

- Clear
- (Delete key)

U
- To clear an object

S
- To undisplay an object

S
- To delete an object's data structures

- Duplicate
- (Ctrl-D)

U
- To duplicate an object

S
- To display an object

S
- To recall an object from the clipboard

S
- To store an object to the clipboard

- Select All
- (Ctrl-A)

U
- To select all objects

S
- To display handles on all objects

- Tool Library...

U
- To substitute a tool for another in the palette

S
- To display the substituted tool in the palette

S
- To record the substitution of one tool for another

U
- To select a tool from the Tool Library

S
- To display the selected tool

U
- To open the Tool Library window

S
- To display the Tool Library window

Arrange Menu

- Bring Forward

U
- To bring an object forward

S
- To move an object higher in the selection hierarchy

- Bring to Front

U • To bring an object to the front

S • To move an object to the top of the selection hierarchy

- Send Backward

U • To send an object backward

S • To move an object lower in the selection hierarchy

- Send to Back

U • To send an object to the back

S • To move an object to the bottom of the selection hierarchy

---

- Group
- (Ctrl-G)

U • To group objects

S • To combine selected objects into a single object

U • To select two or more objects

S • To display handles on two or more objects

- Ungroup
- (Ctrl-Shift-G)

U • To ungroup objects

S • To separate one object into multiple objects

- Show/Hide Grid
- (Ctrl-L)

U • To show (or hide) the grid

S • To display (or undisplay) the grid

- Show/Hide Ruler
- (Ctrl-R)

U • To show (or hide) the ruler

S • To display (or undisplay) the ruler

- Rulers...
- (Ctrl-Y)

U • To set ruler attributes

S • To record ruler attributes

U • To open the Dimensions tab of the Workspace Attributes window

S • To display the Dimensions tab of the Workspace Attributes window

---

- Rotate
- (Ctrl-T)

U • To rotate an object

S • To activate rotation mode

U • To select an object

S • To display an object's handles

- Zoom In
- (Ctrl-E)

U • To zoom in on the design

S • To update the display

S • To increase the zoom factor

- Zoom Out
- (Ctrl-Shift-E)

U • To zoom out of the design

S • To update the display

S • To decrease the zoom factor

Font Menu

- 9; 10; 12; 14; 18; 24

U • To change the font size

S • To update the display of the text object

U • To select a 9 or 10 or 12 or 14 or 18 or 24

S • To record font size selection

- Font selected from cascading menus

U • To change the font

S • To update the display of the text object

U • To select a font

S • To record the font selection

View Menu

- Workspace...
- Double-click in the workspace (see Window section below)

U • To change workspace attributes

S • To record changes in workspace attributes

U • To examine the workspace attributes

S • To display workspace attributes

U • To open the Workspace Attributes window

S • To display the Workspace Attributes window

- Links...

U • To change link display settings

S • To record changes in link display settings

U • To examine the link display settings

S • To display link display settings

U • To open the Link Display window

S • To display the Link Display window

- Object Info...

U • To examine a selected object's attributes

S • To display a selected object's attributes

U • To open the Object Info window

S • To display the Object Info window

- All Objects Info...

U
- To examine the attributes of objects of the type (of object) selected

S
- To display the attributes of all objects of the type (of object) selected

U
- To open the All Objects Info window

S
- To display the All Objects Info window

- Workstation...
- Double-click on a workstation (see Window section below)
- Attributes tab selected in Workstation window

U
- To change workstation attributes

S
- To record changes in workstation attributes

U
- To examine a workstation's attributes

S
- To display workstation's attributes

U
- To open the Workstation window

S
- To display the Workstation window

- Obstruction...
- Double-click on an elemental obstruction (see Window section below)

U
- To change an elemental obstruction's attributes

S
- To record changes in an elemental obstruction's attributes

U
- To examine an elemental obstruction's attributes

S
- To display an elemental obstruction's attributes

U
- To open the Elemental Obstruction window

S
- To display the Elemental Obstruction window

- Fixed Obstruction...
- Double-click on a fixed obstruction (see Window section below)

U
- To change a fixed obstruction's attributes

S
- To record changes in a fixed obstruction's attributes

U
- To examine a fixed obstruction's attributes

S
- To display a fixed obstruction's attributes

U
- To open the Fixed Obstruction window

S
- To display the Fixed Obstruction window

- Other Object...
- Double-click on a generic object (see Window section below)

U
- To change a generic object's attributes

S
- To record changes in a generic object's attributes

U
- To examine a generic object's attributes

S
- To display a generic object's attributes

U
- To open the Other Object window

S
- To display the Other Object window

- Cost Function...

U
- To examine the most recently run cost function

S
- To display the most recently run cost function

U
- To open the Cost Function window

S
- To display the Cost Function window

- Cost Display...

U
- To examine the (colour) cost display(s)

S
- To display the (colour) cost display

U
- To open the Cost Display window

S
- To display the Cost Display window

- Cost Function History...

U
- To recall a design

S
- To display a (recalled) design

U
- To open the Cost Function History window

S
- To display the Cost Function History window

Data Menu

- Link Functions...
- (Ctrl-Shift-L)
- Link Functions tab selected in Workstation window

U
- To change link function data for a workstation

S
- To record changes in link function data for a workstation

U
- To examine link function data for a workstation

S
- To display link function data for a workstation

U
- To open the Link Functions tab of the Workstation window

S
- To display the Link Functions tab of the Workstation window

- Priority Weights...
- (Ctrl-Shift-P)
- Priority Weights tab selected in Workstation window

U
- To change priority weight data for one or two workstations

S
- To record changes in priority weight data for one or two workstations

U
- To examine priority weight data

S
- To display priority weight data for one or two workstations

U
- To open the Priority Weights tab of the Workstation window

S
- To display the Priority Weights tab of the Workstation window

- Domain Weights...
- (Ctrl-Shift-D)

U
- To change domain weights

S
- To record changes in domain weights

U
- To examine the domain weights

S
- To display the domain weights

U
- To open the Domain Weights tab of the Workspace Attributes window

S
- To display the Domain Weights tab of the Workspace Attributes window

---

- Commit All

U
- To commit all priority weights

S
- To set all commit flags to "true"

Execute Menu

- Cost Function
- (Ctrl-F)

U • To examine the cost function
S • To display the cost function
S • To compute the cost function

- Cost Function...

U • To examine the cost function
S • To display the cost function
S • To compute the cost function
U • To open the Cost Function window
S • To display the Cost Function window
U • To set the computation of the cost function to "Automatic" ("Manual")
S • To change the mode of computing the cost function to "Automatic" ("Manual")
U • To save the cost function details to a file
S • To output the cost function details to a file
U • To examine the cost function details
S • To display the cost function details
U • To print the contents of the Cost Function Window
S • To output the contents of the Cost Function Window to the printer

---

- Optimize Workspace

U • To optimise a workspace
S • To open the "Optimize Workspace" Window
S • To invoke the currently implemented optimiser
U • To open the Optimiser Settings window
S • To display the Optimiser Settings window

- Minimize Boundary

U • To minimise the boundary of a selected object
S • To display the size of the boundary as the minimum of the object
S • To compute the minimum size of an object

Window Menu

- Item selected

U • To make the selected window active
S • To display the selected window

Help Menu

- About Locate...

U • To open the Locate splash screen
S • To display the Locate splash screen

- Web Help...

U • To open the Web Help window
S • To display the Web Help window

- Smart Help...

U • To open the Smart Help window

S • To display the Smart Help window

- **LOCATE Actions and Goals by Palette**
  - Selection Arrow

U • To select the Selection Arrow palette item

S • To enter selection mode

  - <Workstation>

U • To create a Workstation

S • To create the data structures for a Workstation

S • To display a workstation in the WS

U • To select the workstation palette item

S • To enter Workstation creation mode

  - <Elemental Rectilinear Obstruction>

U • To create an Elemental Rectilinear Obstruction

S • To create the data structures for an Elemental Rectilinear Obstruction

S • To display an Elemental Rectilinear Obstruction in the WS

U • To select the Elemental Rectilinear Obstruction palette item

S • To enter Elemental Rectilinear Obstruction creation mode

  - <Elemental Elliptical Obstruction>

U • To create an Elemental Elliptical Obstruction

S • To create the data structures for an Elemental Elliptical Obstruction

S • To display an Elemental Elliptical Obstruction in the WS

U • To select the Elemental Elliptical Obstruction palette item

S • To enter Elemental Elliptical Obstruction creation mode

  - <Fixed Rectilinear Obstruction>

U • To create a Fixed Rectilinear Obstruction

S • To create the data structures for a Fixed Rectilinear Obstruction

S • To display a Fixed Rectilinear Obstruction in the WS

U • To select the Fixed Rectilinear Obstruction palette item

S • To enter Fixed Rectilinear Obstruction creation mode

  - <Fixed Elliptical Obstruction>

U • To create a Fixed Elliptical Obstruction

S • To create the data structures for a Fixed Elliptical Obstruction

S • To display a Fixed Elliptical Obstruction in the WS

U • To select the Fixed Elliptical Obstruction palette item

S • To enter Fixed Elliptical Obstruction creation mode

  - <Chair>

U • To create a Chair

S • To create the data structures for a Chair

S • To display a Chair in the WS

U • To select the Chair palette item

S • To enter Chair creation mode

- <Desk>
U - To create a Desk
  S - To create the data structures for a Desk
    S - To display a Desk in the WS
U - To select the Desk palette item
  S - To enter Desk creation mode
- <Console>
U - To create a Console
  S - To create the data structures for a Console
    S - To display a Console in the WS
U - To select the Console palette item
  S - To enter Console creation mode
- <Computer>
U - To create a Computer
  S - To create the data structures for a Computer
    S - To display a Computer in the WS
U - To select the Computer palette item
  S - To enter Computer creation mode
- <Cabinet>
U - To create a Cabinet
  S - To create the data structures for a Cabinet
    S - To display a Cabinet in the WS
U - To select the Cabinet palette item
  S - To enter Cabinet creation mode
- <Column>
U - To create a Column
  S - To create the data structures for a Column
    S - To display a Column in the WS
U - To select the Column palette item
  S - To enter Column creation mode
- <Partition>
U - To create a Partition
  S - To create the data structures for a Partition
    S - To display a Partition in the WS
U - To select the Partition palette item
  S - To enter Partition creation mode
- <Stairway>
U - To create a Stairway
  S - To create the data structures for a Stairway
    S - To display a Stairway in the WS
U - To select the Stairway palette item
  S - To enter Stairway creation mode

- <Pelorus>

U • To create a Pelorus

S • To create the data structures for a Pelorus

S • To display a Pelorus in the WS

U • To select the Pelorus palette item

S • To enter Pelorus creation mode

- <Wall>

U • To create a Wall

S • To create the data structures for a Wall

S • To display a Wall in the WS

U • To select the Wall palette item

S • To enter Wall creation mode

- <Window>

U • To create a Window

S • To create the data structures for a Window

S • To display a Window in the WS

U • To select the Window palette item

S • To enter Window creation mode

- <Door>

U • To create a Door

S • To create the data structures for a Door

S • To display a Door in the WS

U • To select the Door palette item

S • To enter Door creation mode

- <Text>

U • To create a Text object

S • To create the data structures for a Text object

S • To display a Text object in the WS

U • To select the Text object palette item

S • To enter Text creation mode

- <Line>

U • To create a Line

S • To create the data structures for a Line

S • To display a Line in the WS

U • To select the Line palette item

S • To enter Line creation mode

- <Arc>

U • To create an Arc

S • To create the data structures for an Arc

S • To display an Arc in the WS

U • To select the Arc palette item

S • To enter Arc creation mode

- <Ellipse>
U - To create an Ellipse
S - To create the data structures for an Ellipse
S - To display an Ellipse in the WS
U - To select the Ellipse palette item
S - To enter Ellipse creation mode
- <Rectangle>
U - To create a Rectangle
S - To create the data structures for a Rectangle
S - To display a Rectangle in the WS
U - To select the Rectangle palette item
S - To enter Rectangle creation mode
- <Rounded Rectangle>
U - To create a Rounded Rectangle
S - To create the data structures for a Rounded Rectangle
S - To display a Rounded Rectangle in the WS
U - To select the Rounded Rectangle palette item
S - To enter Rounded Rectangle creation mode
- <Polygon>
U - To create a Polygon
S - To create the data structures for a Polygon
S - To display a Polygon in the WS
U - To select the Polygon palette item
S - To enter Polygon creation mode

- **LOCATE Actions and Goals by Windows**

<u>General Window Actions and Goals</u>

- Close button pressed
U - To close the window
S - To undisplay the window
- Window moved
U - To move a window
S - To display a window in a new location
- Window resized
U - To resize a window
S - To display a window with a new size
- Minimize button pressed
U - To minimise a window
S - To display a window in its minimum size
- Maximize button pressed
U - To maximise a window
S - To display a window in its maximum size

CFV Window

- Text area clicked
  - U • To examine a CFV
    - S • To display a CFV
  - U • To save a CFV to the CF History [Manual Mode]
    - S • To output a temporary file
  - U • To compute a CF
    - S • To performing the calculations necessary to generate a CF

Browser Window

- Bounding rectangle repositioned
  - U • To view another part of the design
    - S • To display another part of the design
- Divider between overview and design moved
  - U • To change the relative sizes of the browser and design window
    - S • To display different relative sizes for the browser and design windows

Design Window

- Object clicked
  - U • To select an object
    - S • To display handles on an object
      - S • To add an object to the list of currently selected objects
- Object shift-clicked
  - U • To select multiple objects
    - S • To display handles on selected objects
      - S • To add an object to the list of currently selected objects
- Click-and-drag marquee around object(s)
  - U • To select multiple objects
    - S • To display handles on selected object(s)
      - S • To add an object to the list of currently selected objects
- Object control-clicked
  - U • To select an object inside a workstation or obstruction
    - S • To display handles on an object
- Object Shift+ Control-clicked
  - U • To select multiple objects inside a workstation or obstruction
    - S • To display handles on selected objects inside a workstation or obstruction
      - S • To add an object to the list of currently selected objects inside a workstation or obstruction
- Object clicked-and-dragged
  - U • To move an object
    - S • To display an object in a new position
  - U • To change the coordinates of an object
    - S • To update the X and/or Y coordinates of an object

- Object handle clicked-and-dragged

U • To resize an object

S • To display an object with a new size

U • To change the dimensions of an object

S • To update the dimensions of an object

- Object clicked-and-dragged when in rotate mode

U • To rotate an object

S • To display an object in a newly rotated position

U • To change the angle of an object

S • To update the angle of an object

- Mouse clicked in the workspace after choosing a tool in the palette (other than the Selection Arrow)

U • To create a default-sized object with the specified position and angle

S • To display an object of default size with specified position and angle

S • To create data structures for an object of default size with specified position and angle

- Mouse clicked-and-dragged in the workspace after choosing a tool in the palette (other than the Selection Arrow)

U • To create an object with specified size, position and angle

S • To display an object with specified size, position and angle

S • To create data structures for an object with specified size, position and angle

- Click in empty space

U • To deselect selected objects

S • To undisplay the handles on selected objects

S • To remove all objects from the list of selected objects

- Double-click in empty space
  - see Menu section above: Workspace Attributes
- Double-click on object
  - see Menu section above: EW/EOb/FOb/OO Attributes

<u>Object Info Window</u>

- No actions other than basic window actions

<u>All Objects Info Window</u>

- Update button pressed

U • To update the All Objects Info window

S • To display updated information in the All Objects Info window

Workspace Attributes

- Text changed in a text box
  - U • To change workspace attributes
    - S • To record changes to workspace attributes
- OK button pressed
  - U • To accept any attribute changes to the workspace
    - S • To display any attribute changes to the workspace
      - S • To undisplay the workspace attributes window
        - S • To record any accepted attribute changes to the workspace
- Cancel button pressed
  - U • To reject any attribute changes to the workspace
    - S • To undisplay the workspace attributes window
      - S • To cancel any attribute changes to the workspace
  - U • To close the window
    - S • To undisplay the window
- Apply button pressed
  - U • To (possibly) revise changes to the workspace
    - S • To leave the workspace attributes window open
  - U • To accept any attribute changes to the workspace
    - S • To display any attribute changes to the workspace
      - S • To record any accepted attribute changes to the workspace

Workstation

- Attributes tab selected
  - • see Menu section above: View Menu, Workstation
- Link Functions tab selected
  - • see Menu section above: Data Menu, Link Functions
- Priority Weights tab selected
  - • see Menu section above: Data Menu, Priority Weights
- Text changed in the x-position text box
  - U • To change the x-position of the workstation
    - S • To record the change to the x-position of the workstation
- Text changed in the y-position text box
  - U • To change the y-position of the workstation
    - S • To record the change to the y-position of the workstation
- Text changed in the angle text box
  - U • To change the angle of the workstation
    - S • To record the change to the angle of the workstation
- Text changed in the x-position minimum text box
  - U • To change the x-position minimum of the workstation
    - S • To record the change to the x-position minimum of the workstation
- Text changed in the y-position minimum text box
  - U • To change the y-position minimum of the workstation
    - S • To record the change to the y-position minimum of the workstation

- Text changed in the angle minimum text box

U • To change the angle minimum of the workstation

S • To record the change to the angle minimum of the workstation

- Text changed in the x-position maximum text box

U • To change the x-position maximum of the workstation

S • To record the change to the x-position maximum of the workstation

- Text changed in the y-position maximum text box

U • To change the y-position maximum of the workstation

S • To record the change to the y-position maximum of the workstation

- Text changed in the angle maximum text box

U • To change the angle maximum of the workstation

S • To record the change to the angle maximum of the workstation

- Text changed in the radius text box

U • To change the radius of the workstation

S • To record the change to the radius of the workstation

- Text changed in the S/R x-position text box

U • To change the x-position of the S/R of the workstation

S • To record the change to the x-position of the S/R of the workstation

- Text changed in the S/R y-position text box

U • To change the y-position of the S/R of the workstation

S • To record the change to the y-position of the S/R of the workstation

- Text changed in the S/R angle text box

U • To change the angle of the S/R of the workstation

S • To display the change to the angle of the S/R of the workstation

- Selection of radio button in the link function matrix

U • To change the attributes of a link function component of a workstation

S • To record changes in a link function component of a workstation

U • To examine the attributes of a link function component of a workstation

S • To change link function component modification mode for a workstation

- Selection of a link function in the pop-up menu

U • To specify a link function for a link function component of a workstation

S • To display the graph for the selected link function in the window

S • To display the name of the selected link function in the pop-up menu

S • To record the selected link function for the selected link function component

- Text changed in a link function argument text box
- Handles dragged in the function graph
- Exponent buttons pressed in the function graph

U • To modify a link function argument for a selected link function component for a workstation

S • To update the link function graphs

S • To record the link function argument

- Text changed in a priority weight text box

U • To modify the priority weights for a (pair of) workstation(s)

S • To display the modified text

- OK button pressed
U - To accept any attribute changes to the workstation
S - To display any attribute changes to the workstation
S - To undisplay the workstation attributes window
S - To record any attribute changes to the workstation
- Cancel button pressed
U - To reject any attribute changes to the workstation
S - To undisplay the workstation attributes window
S - To cancel any attribute changes to the workstation
- Apply button pressed
U - To (possibly) revise changes to the workstation
S - To leave the workstation attributes window open
U - To accept any attribute changes to the workstation
S - To display any attribute changes to the workstation
S - To record any accepted attribute changes to the workstation

Elemental Obstruction

- Text changed in a text box
U - To change the obstruction's attributes
S - To display changes to obstruction's attributes
- Exponent buttons pressed in the function graph
U - To modify the obstruction's function arguments
S - To update the obstruction's function graphs
- OK button pressed
U - To accept any attribute changes to the obstruction
S - To display any attribute changes to the obstruction
S - To undisplay the obstruction attributes window
S - To record any accepted attribute changes to the obstruction
- Cancel button pressed
U - To reject any attribute changes to the obstruction
S - To undisplay the obstruction attributes window
S - To cancel any attribute changes to the obstruction
- Apply button pressed
U - To (possibly) revise changes to the obstruction
S - To leave the obstruction attributes window open
U - To accept any attribute changes to the obstruction
S - To display any attribute changes to the obstruction
S - To record any accepted attribute changes to the obstruction

Fixed Obstruction

- Text changed in a text box
  - U • To change the obstruction's attributes
    - S • To display changes to obstruction's attributes
- Exponent buttons pressed in the function graph
  - U • To modify the obstruction's function arguments
    - S • To update the obstruction's function graphs
- OK button pressed
  - U • To accept any attribute changes to the obstruction
    - S • To display any attribute changes to the obstruction
      - S • To undisplay the obstruction attributes window
        - S • To record any accepted attribute changes to the obstruction
- Cancel button pressed
  - U • To reject any attribute changes to the obstruction
    - S • To undisplay the obstruction attributes window
      - S • To cancel any attribute changes to the obstruction
- Apply button pressed
  - U • To (possibly) revise changes to the obstruction
    - S • To leave the obstruction attributes window open
  - U • To accept any attribute changes to the obstruction
    - S • To display any attribute changes to the obstruction
      - S • To record any accepted attribute changes to the obstruction

Other Object

- Text changed in a text box
  - U • To modify the object's attributes
    - S • To display changes to object's attributes
- OK button pressed
  - U • To accept any attribute changes to the object
    - S • To display any attribute changes to the object
      - S • To undisplay the object attributes window
        - S • To record any accepted attribute changes to the object
- Cancel button pressed
  - U • To reject any attribute changes to the object
    - S • To undisplay the object attributes window
      - S • To cancel any attribute changes to the object
- Apply button pressed
  - U • To (possibly) revise changes to the object
    - S • To leave the object attributes window open
  - U • To accept any attribute changes to the object
    - S • To display any attribute changes to the object
      - S • To record any accepted attribute changes to the object

CF History

- List item double-clicked
  - List item selected and Recall Design button pressed

U - To recall a design

S - To display a design

- Clear All Entries button pressed

U - To clear the Cost Function history

S - To delete all Cost Function History temporary files

- Within Designs radio button pressed

U - To list the cost function values within designs

S - To display the cost function values in within designs mode

- Across Designs radio button pressed

U - To list the cost function values across designs

S - To display the cost function values in across designs mode

- Sort pop-up menu item selected

U - To change the sorting of cost function values

S - To re-sort the list of cost function values according to the new criteria

Cost Display

- Link Quality checkbox pressed

U - To display link qualities in the colour cost display

S - To enter link quality mode in the colour cost display

- Weighted Cost checkbox pressed

U - To display weighted costs in the colour cost display

S - To enter weighted costs mode in the colour cost display

- Normalized checkbox pressed
- Non-normalized checkbox pressed

U - To display normalised (non-normalised) costs in the colour cost display

S - To enter normalised (non-normalised) costs mode in the colour cost display

- Display Editor button pressed

U - To change the criteria of the colour cost display

S - To record changes to the criteria of the colour cost display

U - To open the Cost Editor window

S - To display the Cost Editor window

- Print button pressed

U - To print the cost display

S - To output the cost display to the printer

- Close button pressed

U - To close the currently active Cost Display window

S - To undisplay the currently active Cost Display window

- Close All button pressed

U - To close all of the Cost Display windows

S - To undisplay all of the Cost Display windows

- Window pop-up menu item selected

U - To open to the specified window

S - To display the specified window

Cost Function

- Manual Calculation checkbox pressed
  - U • To specify manual cost function computation
    - S • To update the display of the checkbox
      - S • To enter manual cost function computation mode
- Automatic Calculation checkbox pressed
  - U • To specify automatic cost function computation
    - S • To update the display of the checkbox
      - S • To enter automatic cost function computation mode
- Add Details to Output File checkbox pressed
  - U • To specify creation of an output file
    - S • To update the display of the checkbox
      - S • To enter add details to output file mode
- Display Details checkbox pressed
  - U • To specify display of cost function details
    - S • To update the display of the checkbox
      - S • To enter display details to output file mode
- Print button pressed
  - U • To print the cost function information
    - S • To output the cost function information to the printer
- Run (Again) button pressed
  - U • To run the cost function
    - S • To compute and display the cost function
- OK button pressed
  - U • To accept the changes to the settings in the Cost Function Window
    - S • To record the changes to the settings in the Cost Function Window
  - U • To close the window
    - S • To undisplay the window

Tool Library

- Tool pop-up menu item selected
  - U • To change the current tool
    - S • To update the display of the pop-up menu
- OK button pressed
  - U • To accept any change in the tool
    - S • To undisplay the Tool Library window
      - S • To record any change in the tool
- Cancel button pressed
  - U • To reject any change in the tool
    - S • To undisplay the Tool Library window
      - S • To cancel any changes in the tool

- Apply button pressed
  - U • To (possibly) revise the substitution for the selected tool
  - S • To leave the Tool Library window open
  - U • To accept any change in the tool
  - S • To display the substituted tool in the palette
  - S • To record any change in the tool

Link Display

- Link checkbox pressed
  - U • To change the display of links
  - S • To update the display of the checkbox
  - S • To record the change in the display of links
- OK button pressed
  - U • To accept any changes in the display of links
  - S • To update the design window
  - S • To undisplay the Link Display window
  - S • To record any changes in the display of links
- Cancel button pressed
  - U • To reject any changes in the display of links
  - S • To undisplay the Link Display window
  - S • To cancel any changes to the Link Display window
- Apply button pressed
  - U • To (possibly) revise the changes to the display of links
  - S • To leave the Link Display window open
  - U • To accept any changes in the display of links
  - S • To update the design window
  - S • To record any changes in the display of links
  - U • To examine the effects of the changes in the workspace?
  - S • To update the design window

About LOCATE

- OK button pressed
  - U • To close the window
  - S • To undisplay the window
- More on LOCATE button pressed
  - U • To view additional information on LOCATE
  - S • To display the More on LOCATE Window
  - S • To undisplay the About LOCATE window

Smart Help

- Text changed in the Name text box
  - U • To provide a user identifier to LOCATE
  - S • To display the identifier text
- Checkbox pressed
  - U • To change the help options
  - S • To display the modified text

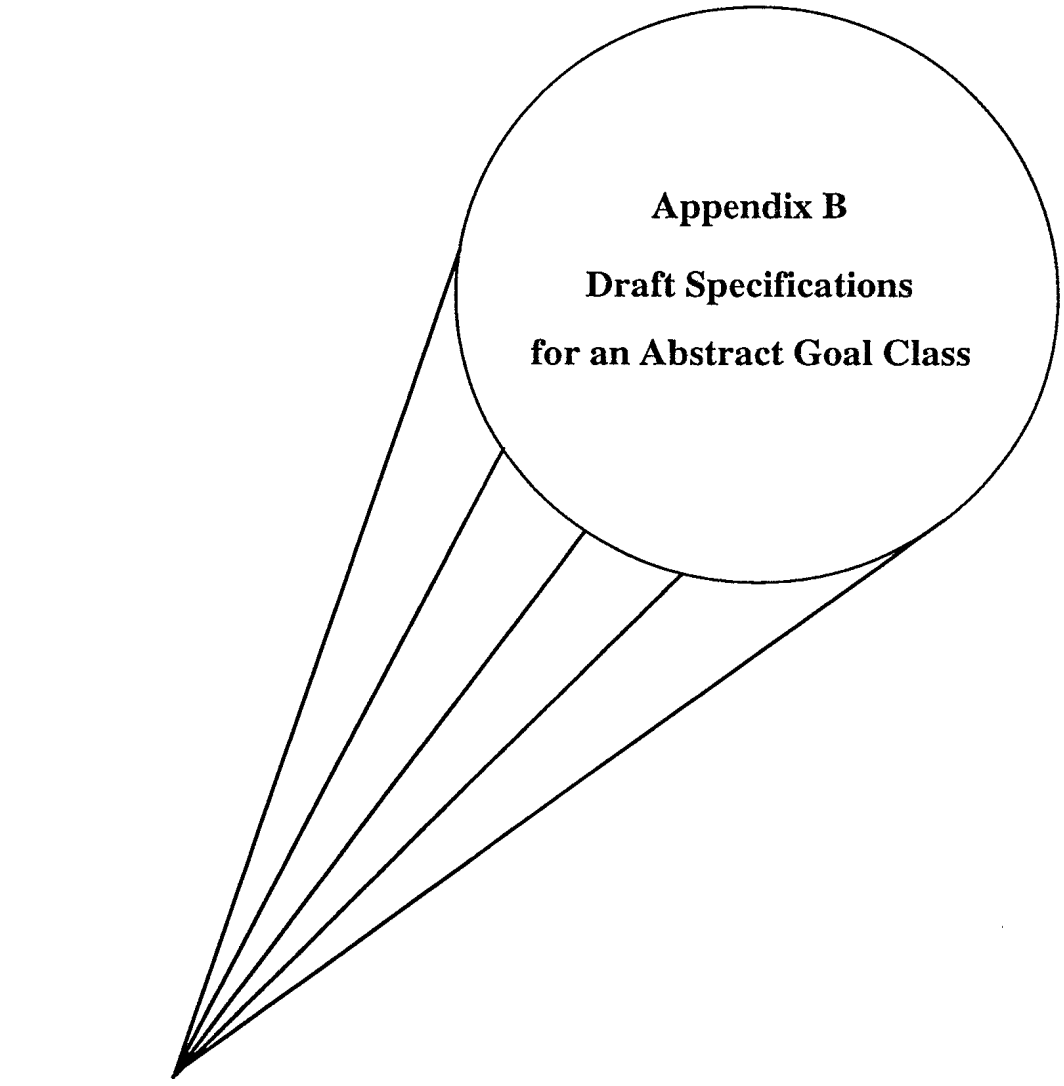
- OK button pressed
  - U • To accept any changes to Smart Help attributes
  - S • To undisplay the Smart Help window
  - S • To record any changes to Smart Help attributes
- Cancel button pressed
  - U • To reject the Smart Help changes, if any
  - S • To undisplay the Smart Help window
  - S • To cancel any changes to the Smart Help window

Print Preview

- Preview area clicked or clicked and dragged
  - U • To specify the portion of the workspace to print
  - S • To display the portion of the workspace to print
- OK button pressed
  - U • To accept any changes to the page ranges to be printed
  - S • To undisplay the Print Preview window
  - S • To record any changes to page ranges to be printed
- Cancel button pressed
  - U • To reject any changes to the page ranges to be printed
  - S • To undisplay the Print Preview window
  - S • To cancel any changes to the Smart Help window
- Print button pressed
  - U • To print the design
  - S • To output the specified page ranges of the design to the printer
  - S • To undisplay the window

Appendix B

Draft Specifications for an Abstract Goal Class



Artificial Intelligence
Management and Development Corporation

On the following pages is a draft set of specifications for an abstract goal class for LOCATE. The specifications are expressed in a form consistent with the object-oriented paradigm and contain a variety of items and notes about how that class is to be realised.

Currently, a small subset of this goal class is instantiated whenever the user takes some action at the interface. Thus, many instances of goals are created, satisfied or abandoned, and disappear during a given LOCATE session. Remnants of those goal objects are retained in the Action and Goal History portion of LOCATE's Task Model Window.

To get an idea of the type and number of those goals, see Appendix B, above.

specification Goal (Abstract Class)

SuperClass
None

SubClasses
Goal
- TaskGoal
  - GeneralGoal
    - ToProvideInfo
  - DesignGoal
    - ToAdd(Design)
    - ToShow(Design)
    - ToHide(Design)
    - ToSelect(Design)
    - ToChange(Design)
    - ToExecute(Design)
    - ToCalculate(Design)
    - ToImport(Design)
    - ToExport(Design)
  - MetaDesignGoal
    - ToAdd(Meta)
    - ToCalculate(Meta)
    - ToChange(Meta)
    - ToGetInfo(Meta)
    - ToDisplayInfo(Meta)
    - ToExecute(Meta)
    - ToHide(Meta)
    - ToSelect(Meta)
    - ToShow(Meta)
- UserGoal
- SelfGoal
- Comm.Goal
  - DialogueGoal
    - InformGoal
    - RequestGoal
      - ToCommand
      - ToRequestDirectly
      - ToRequestIndirectly
    - FeedbackGoal
      - ConfirmatoryGoal
        - ToConfirmNormal
        - ToConfirmNull
      - DisconfirmatoryGoal
        - ToClarify
        - ToCorrect

NonDialogueGoal

notes

Constraints:

1. Goals are constrained by the number of supergoals and subgoals that they can access in the complex of goals of which they are a part.
2. Enabling and disabling conditions constrain when a goal can be pursued, and succesConditions constrain when the goal may be said to be satisfied.
3. There may be several holders and pursuers of a goal. Often, there will be more than one holder of a goal (User and System), but only one pursuer (User or System), especially of the subgoals necessary to reach the a high-level goal.
4. Goal Pursuit: The System would begin with a collection of persistent Goal instances, which it would pursue repeatedly, from the time it became active, either in a cycle or in parallel. Each of those goal instances would, from time to time, create new subgoal instances that would be added to a priority queue of currently active goals. Highest-priority subgoals would, in turn, be pursued and would generate further subgoals until the subgoal(s) gave rise to System actions that did not involve creation of further subgoals. Both satisfied and cancelled goals could be kept as objects in a history-of-goals database to provide the basis of dialogue with the User about past behaviour, and as a context for future System behaviour. [An interesting question is: How much goal history should be kept?]
[Notice the similarity between goals, as described here, and procedures in a procedural language. Goal classes correspond roughly to procedure definitions, and goal instances correspond roughly to the activations of those procedures. But the scheme described here allows more explicit control, inference and explanation than procedural languages provide.]
5. A proposal: Goals would keep a list of all of the subgoals they create, as well as all of the subgoals created by the subgoals below them. Thus, the highest-level goal (or, highest supergoal) contains a list of all the subgoals required to satisfy itself. In contrast, since people often forget how the achieved a goal through some subgoal hierarchy, it may be necessary, in modelling the human user, to have a method of decay of the subgoal structures.

descriptiveAttribute

Class attributes (C&Y's definitionData??)

ComplimentaryGoal: A goal adopted by one partner to support the achievement of a goal by another. In the present case it is the System that is modelled as having complimentary goal to support a User's goal.

AllPossibleGoals: A list of all possible, legitimate goal classes.

PossibleSupergoals: A list of all possible legitimate supergoal classes for a goal; values to be specified by subgoal class

PossibleSubgoals: A list of all possible legitimate subgoal classes for a goal; values to be specified by subgoal class

AllPossibleGoalObjects: A list of all possible, legitimate object classes of a goal; values to be specified by subgoal class

Instance attributes

complimentaryGoal: The current complimentary goal
goalType: The type of goal
how: a reference to the corresponding act
goalDescription: A description of a goal in English, for purposes of explanation; values to be specified by subgoal class
goalType: The name of the Class of the goal; high-level legitimate values are: Task, User, Dialogue and Self
supergoals: A list of the current supergoal(s) of a goal, that is, the supergoals that created the current goal; values specified at runtime (Note: The supergoals may have a different holder that the current goal, e.g., when the system adopts a goal held by User, the goal held by the User might be represented as a supergoal of the one adopted by the System; also, these supergoals may be inferred to be ones held by the User based on the current goal
subgoals: An ordered collection of the current subgoals of a goal; values specified at runtime
persistent:? A boolean indicating whether the goal is a persistent goal (default value is false.
holder(s): A list of agents (e.g., User/System) that are holders of a goal
pursuer(s): A list of agents (e.g., User/System) that are actively attempting to achieve a goal
predicate(s): A list of Act instances
enablingCondidtion(s): A set of conditions necessary before the achievement of a goal is possible
disablingCondition(s): A set of conditions that prevent a goal from being achieved.
successCondition(s): A set of conditions sufficient for the successful achievement of a goal; likely, a boolean combination of subgoals
status: The current status of a goal; values determined at runtime; legitimate values are: satisfied (all successConditions have been met); unsatisfied (currently being pursued); inactive (unsatisfied, not being pursued); or, cancelled (unsatisfied, but no longer to be pursued)
isSatisfied: a boolean (True/False) indicating whether the goal is currently satisfied.
minTime: a time measure indicating the minimum time that must elapse before the goal can be pursued.
maxTime: a time measure indicating the maximum amount of time that can elapse before the goal must be pursued.
purgeTime: a time measure indicating the amount of time that must elapse before the goal is purged, i.e., no longer eligible to be pursued.
invokedBy: a text description of how the goal was invoked

alwaysDerivableAttribute

occasionallyDerivableAttribute

inheritedAttributes:

None

externalSystemInput -
LogFile: The user should be provided with an audit trail mechanism. The audit trail shall allow the user to backtrack to previous configurations, and to determine the heredity of his current database. (RS: p. 7)
Database: Contains a description of the link relationships, the object orientations, and possibly object details such as dimensions, graphic icon representation, etc.. The portions of the database associated with each object should be individually accessible, allowing the user simpler comprehension and control. (RS: p. 7)
Optimiser: An set of optimization algorithms that can be invoked, halted and resumed by a user. (RS: p. 8)
CAD package: "...it is reasonable to expect that users will want to exchange data between their CAD package and LOCATE. Hence, LOCATE should have a facility to import from and export to CAD packages." (RS: p.7)

externalSystemOutput -
LogFile: (Same as above)
Database: (Same as above)
Optimiser: (Same as above)
CAD package: (Same as above)

instanceConnectionConstraint
with User: -
with Elemental Workstation: -
with Auxiliary Object: -
with Log File: -
with Optimiser: -
with CAD System: -

stateEventResponse <...>

objectLifeHistory
ADD

CHANGE ATTRIBUTES (goalDescription; supergoals; subgoals; holder(s); pursuer(s); predicate(s); enablingConditions; disablingConditions; sucessConditions; status; isSatisfied; minTime; maxTime; purgeTime)

COMPUTE

COUNT

SELECT

DISPLAY ATTRIBUTES (goalDescription(S); goalType (Class) (S); AllPossibleGoals (L); PossibleSupergoals (L); PossibleSubrgoals (S);

AllPossibleObjects (L); current superGoals (L); current subGoals (L); peristent (S); holder(s) (L); pursuer(s) (L); predicate(s) (S); enablingCondition(s) (L); disablingConditions (L); successConditions (L); status (S); isSatisfied (S); minTime (N); maxTime (N); purgeTime (N))

RETRIEVE

PRINT

IMPORT/EXPORT (Database(s))

DELETE

intent/purpose

The purpose of the Goal Class is to provide a foundation for representing explicitly to the User and the System itself, what goals the User and the System are pursuing and how they are going about achieving them. The "how" is the means of achieving goals, both in terms of subgoals at the same and different levels of abstraction.

Class operations (Class methods):

These would include operations to create goal instances and to access and possibly to update the class attributes.

Instance operations (services or instance methods):

New
myComplimentaryGoal
mySupergoal
getHow
setSupergoal
mySubgoal
setHolder
setPursuer
addSubgoal
<<>>

Other existing Services: (all not functional)

generate
pursue
cancel
whilePursuing
satisfied?
suspend
whenSatisfied
setSupergoal
setPursuer

setHolder
testSuccess.Perform
- on change in successConditions, will test whether all successConditions are present (When the successCondition is a predicate on the goalObject, this will usually involve operating on the System's model of the goalObject to determine whether the System believes the condition has been satisfied. When the successCondition is a Boolean combination of subgoals, it will involve performing testSuccess operations on the subgoals.)
- will output a message to Sender (Dialogue Model?) as to whether the successCondition currently evaluates to "true".

service pursueGoal.Perform
- will check arguments against the corresponding Definition Attribute constraints (enabling and disabling conditions)
- will change status to "unsatisfied" until successConditions evaluates to true
- will create subgoals required for the satisfaction of the goal; responsibility of specific goal
- will monitor successConditions for satisfaction
- will change isSatisfied condition to true when all successConditions are found to be true

service suspendGoal.Perform
- if goal is "unsatisfied", will make status "cancelled"
- will send cancel messages to subgoals

service cancelGoal.Perform
- if goal is "unsatisfied", will make status "inactive"
- will send suspend messages to subgoals

service cancelGoal.Perform
- if goal is "unsatisfied", will make status "inactive"
- will send suspend messages to subgoals

inheritedServices
None

[Note: Other attributes and services supporting a second-party point-of-view, e.g.,
believedSuccessCondition -- a proposition used to
infer whether the holder will believe the
goal is satisfied.
believedStatus -- inactive, unsatisfied, satisfied, or
cancelled.
believedSubgoals
believedSupergoals

testBelievedSuccess -- test whether the
believedSuccessCondition is currently true.

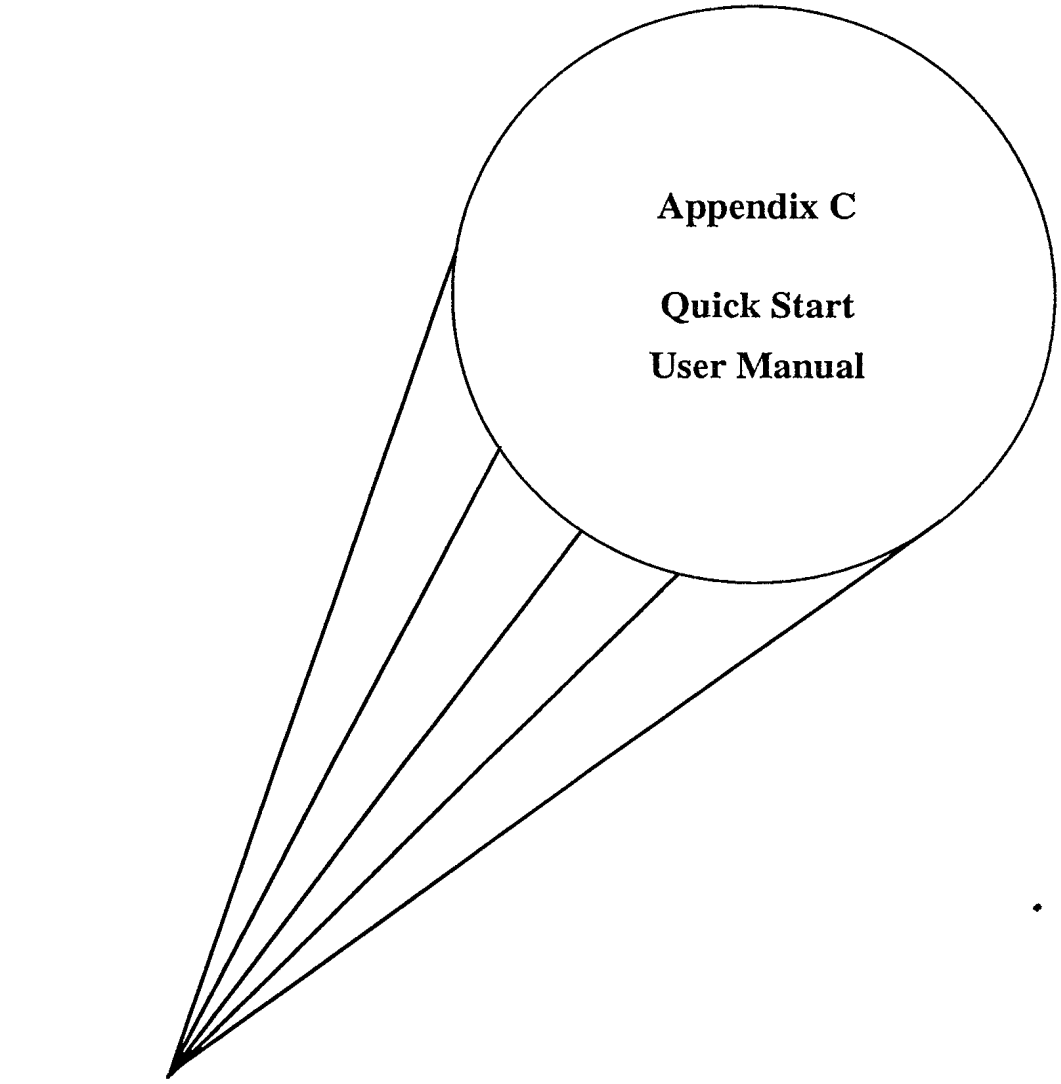
The following operations are used by the System to model relationships between goals attributed to a User.

inferSubgoals -- infer and (if necessary) create
instances of UserGoals that are subgoals of
this goal.
inferSupergoals -- infer and (if necessary) create
instances of UserGoals that are supergoals of
this goal.]

end specification

Appendix C

Quick Start
User Manual



Artificial Intelligence
Management and Development Corporation

Quick Start User Manual

for the

LOCATE Workspace Layout Tool

Table of Contents

INTRODUCTION

LOCATE is three tools in one:

- a Graphical User Interface (GUI) for creating workspace or facility layout designs;
- a program for evaluating the communication efficiency of designs;
- a tool for direct access to the World Wide Web and, thus, to worldwide design information and expanded opportunities for sharing ideas and solving design and workspace problems.

LOCATE's Graphical User Interface

LOCATE's GUI is a relatively recent development and has been constructed so that much of the data required for a LOCATE analysis can be provided through direct manipulation of graphical objects.

As a designer, you are able to enter size, position and orientation data for a given LOCATE object simply by selecting the appropriate tool from LOCATE's palette and clicking and dragging in the workspace.

The LOCATE Program

The LOCATE program allows you to input information about the location, orientation and structure of workstations in a workspace and, when satisfied with a configuration, to have the program evaluate it, or its components, through the generation of Cost Functions. Different configurations may be tried, and their cost functions compared, as a way of helping you determine the "best" design.

The cost function values are a measure of the communication efficiency of your design. LOCATE analyses efficiency for any combination of four modalities:

- Auditory;
- Distance;
- Tactile;
- Visual

Although the LOCATE program was developed for workstation layout, it may be generalised to many other areas. Office and manufacturing plant layouts, ship's bridges, command posts, aircraft cockpits and computer screens are a few of the areas to which it might be applied.

LOCATE Access to the World Wide Web (WWW)

In addition to its primary purpose of design creation and analysis, LOCATE provides you with access to the World Wide Web through a special built-in Web Browser whose home page is a Web site that contains help information describing the benefits and features of LOCATE.

A variety of WWW Browser Bookmarks point you to various Human Factors Sites and information potentially relevant to your designs and analyses.

Of course, access to LOCATE help information on the Web is available through standard Browsers like Netscape and Microsoft Explorer, but access through LOCATE's own Browser is directly available by selecting "Web Help" from the Help Menu.

WORKING WITH LOCATE

Designing a Workspace

Designing a workspace in LOCATE is easy.

Use LOCATE'S tool palette to configure your workspace by clicking on the icon of the tool you want to use and then clicking and dragging in the workspace to create the object.

To find out the name of the tool you have selected, look in the name box at the top of the palette.

Double-clicking on a tool allows you to create as many instances of an object as you like without having to click repeatedly on its palette icon. When you're done, just click on the next tool you wish to use, or on the selection arrow.

Analysing a Design

To run a LOCATE analysis on a design you have created, three pieces of information are necessary:

- information about the location, size, position and angle of rotation of workstations, which is automatically provided when you create them in the workspace.

  Precise information about such characteristics can be specified in the workstation attributes window. Open the attributes windows of any object by double-clicking on the object, or by clicking to select the object and then selecting the appropriate name (e.g., "Workstation") in LOCATE's View Menu.

- information about which communication domains you wish to analyse (auditory; distance; tactile; visual).

To specify domains, open the Domain Weights portion of the Workspace Attributes Window and select any combination of the four domains listed there. That portion of the Workspace Attributes Window can be opened by selecting "Domain Weights" from LOCATE's Data Menu;

- commitment of priority weight default values by selecting "Commit All" from the LOCATE Data Menu, or by performing a similar selection in LOCATE's Cost Function Checks Window, which appears when you try and run a cost function without having committed those values.

To run a cost function, which is LOCATE's measure of communication efficiency, click in the small window at the bottom of the LOCATE tool palette or select "Cost Function" from LOCATE's Execute Menu. The resulting cost function value appears in the same window at the bottom of the palette.

If you want more details about LOCATE analyses, open the Cost Function Window by selecting "Cost Function..." from the Execute Menu and clicking on the two checkboxes at the bottom left of the window. Click the "Run Again" button in the middle at the bottom of that window to re-run the cost function and display more cost details.

Other Data Necessary for Analysis

Default values for two key types of data have been provided to allow you to run a LOCATE analysis almost immediately after you have populated a workspace.

The two sets of default data answer the following questions, respectively:

- who communicates with whom and how (Link Function Data)
- how important is communication between the various elements in your design (Priority Weight Data).

Although the defaults for Link Function and Priority Weight data make it easy to quickly generate a cost function value for your design, they are included only as rough guides for what those values should be for your own particular design.

Link Function Data. Link Function data is provided separately for each workstation in your workspace.

To enter Link Function Data, select a workstation and double click to open its attributes window and then click the "Link Functions" tab. Alternatively, click on a workstation and select "Link Functions" from the Data Menu.

Choose from a list of several available functions as a way of describing the communication characteristics of the workstation. Different functions may be used for specifying the distance and angular components for each communication domain that LOCATE has been asked to analyse and, that is done for each workstation, considered as both a source and receiver of information.

Therefore, for each workstation, considered as a source and receiver (2), for each distance and angular component (2), for all four domains (4) , you would need to specify sixteen (2 x 2 x 4) functions and their arguments.

For help in deciding what functions and arguments to use, refer to LOCATE's hypertext help files.

Priority Weight Data. Priority Weight data is specified for **each pair of workstations** in your workspace.

To enter Priority Weight data, select any pair of workstations and then select the "Priority Weights" item from LOCATE's Data Menu.

In the Priority Weights Window, each Workstation is identified first as the **source** of information, with the second workstation as the **receiver**, and then as the **receiver**, with the second workstation as the **source**.

Data are entered for the workstations for each communication domain being analysed.

Entries range from –1 to +1, indicating the importance of a given type of communication between the two workstations. Positive values indicate that it is more or less desirable for

one or both of the pair of workstations to communicate with each other, while negative values indicate that such communication is more or less undesirable.

For help in deciding how to specify priority weights, refer to LOCATE's hypertext help files.

Refining Designs

Once a design has been created, you may run a cost function to determine the efficiency of the design. To run a cost function, see the relevant portion of the section above entitled, "Analysing a Design."

Once you have run a cost function, you may change the design in any of several different ways, e.g., by changing positions, sizes and rotation angles of objects.

Once you have made those changes, re-run the cost function to determine any change in efficiency.

Comparing Designs

Designs containing identical workstations may contain other, different objects such as chairs, partitions, platforms, etc.. If you are working with more than one such design, you can run cost functions for each and compare the results to determine which is the most efficient. (Again, see the earlier section, entitled, "Analysing a Design" for instructions on how to run a cost function.)

Each time you run a cost function, LOCATE saves the design configuration in a temporary file. That design may be recalled at a later time using LOCATE's **Cost Function History Window.**

To display the Cost Function History Window, select, "Cost Function History" from the View Menu. The resulting window contains a list of all cost function values generated during a session. By double-clicking on any one of those values, you are able to redisplay its associated design in the workspace.

By default, cost function values are sorted by the order of their creation within each design that you have opened (or created) during a session, however, you also may display those values in ascending or descending order or, in any of the three sort orders across (as opposed to within) designs.

Saving a Recalled Design. At any time, you may save a design you have recalled. Using the Cost Function History Window, simply recall the design, click in the Design Window and then select, "Save" from the File Menu.

Since it is unlikely that the last configuration on the screen will be your most efficient one, the Cost Function History Window is an important LOCATE feature for preserving your optimal configuration.

You may clear all cost function values from the Cost Function History Window by selecting, "Clear All Entries." Be aware, however, that if you do, all the (temporary) design files associated with those entries will be lost.

Optimising Designs

In addition to analysing and comparing designs, LOCATE provides a build-in optimiser[†] that automatically optimises a set of workstations. Although that optimiser is a simple one, it provides a fair demonstration for how optimisation can work within LOCATE. In spite of its simplicity, the optimiser has been used successfully in a recent practical application of the LOCATE tool.

After you have created your design, have LOCATE optimise it by selecting, "Optimize Workspace" from the Execute Menu.

LOCATE then displays a window in which you choose one of two methods of optimisation: a) swapping positions of workstations; or, 2) moving workstations about in the workspace and changing their angles of rotation. You may alter the default values for the step size in units and the number of degrees by which each workstation will be moved or rotated, respectively, during optimisation.

[†] The AIM Optimiser© was developed by Artificial Intelligence Management and Development Corporation, 206 Keewatin Ave., Toronto, Ontario, Canada M4P 1Z8.

Other options include instructing LOCATE to keep certain workstations together and others apart, as optimisation proceeds.

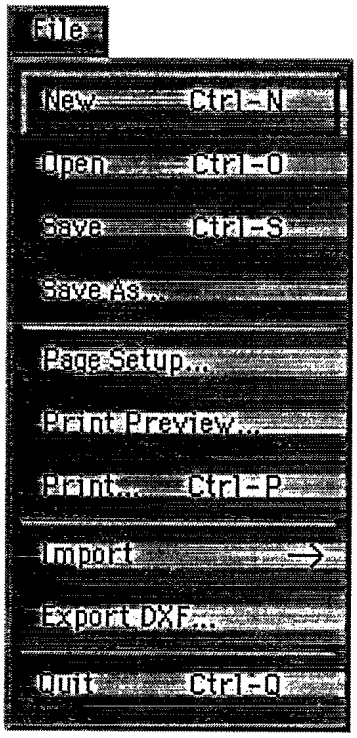
Once you are satisfied with your optimisation settings, click "Optimize" to begin. The positions and angles of workstations are updated on the screen every five seconds and you may pause, stop or abort the optimisation at any point during the process.

Also, at any time after you have paused or stopped an optimisation, you may save the cost function value to the cost function history for recall at a later time or, you may save the design as a permanent file.

LOCATE'S MENUS

The following shows each of LOCATE's Menus and provides a brief comment about the items in those Menus.

The File Menu



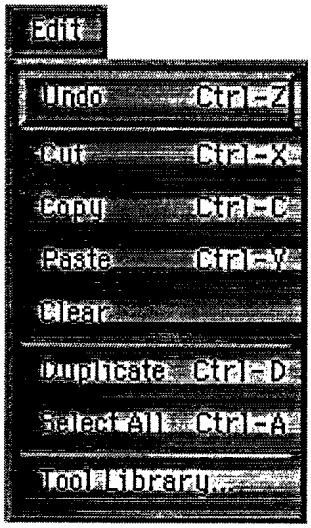
New – Creates a new workspace.
Open – Opens an existing workspace.
Save – Saves the current workspace.
Save As... – presents a standard dialogue window for saving a design.

Setup – presents a standard page setup window for specifying page attributes.
Print Preview – presents a panel matrix for specifying the area(s) of the design you want to print.
Print– presents a standard print dialogue box for specifying print attributes and printing a design.

Import → – a cascading menu with two submenu choices: 1) "DXF file"; and, 2) "Data." The first imports a standard DXF file; the latter imports link function and priority weight data from another design, so long as the number and names of workstations correspond to the current design.
Export DXF – exports a LOCATE design in DXF format.

Quit – quits LOCATE.

The Edit Menu



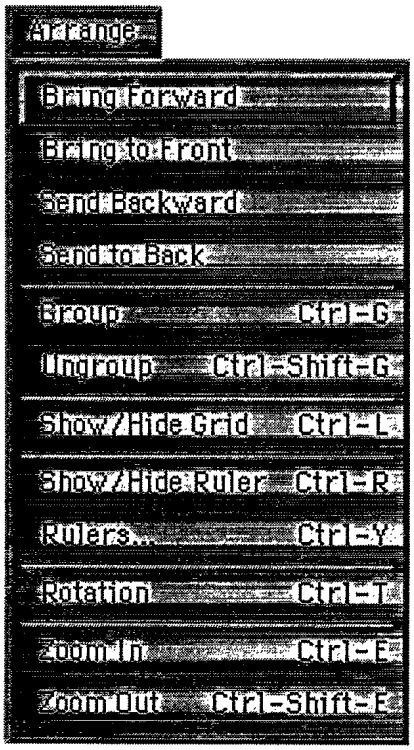
Undo – Undoes the last command, with some exceptions.

Cut – cuts the currently selected object or text.
Copy – copies the currently selected object or text.
Paste – pastes the object or text that has been copied or cut.
Clear – clears the currently selected object or text.

Duplicate –duplicates the currently selected object or text.
Select All – selects all objects and text in the workspace.

Tool Library – opens a tool library that allows you to substitute one palette item for the currently selected palette item. The exception is the selection arrow, which cannot be substituted.

The Arrange Menu



Bring Forward – brings the currently selected object forward relative to other objects.
Bring to Front - brings the currently selected object to the front of other objects.
Send Backward – sends the currently selected object backward relative to other objects.
Bring to Back - sends the currently selected object to the back of other objects.

Group – groups currently selected objects together as one object, with some exceptions.
Ungroup – ungroups the currently selected grouped object.

Show/Hide Grid – a toggle command that shows or hides the workspace grid.

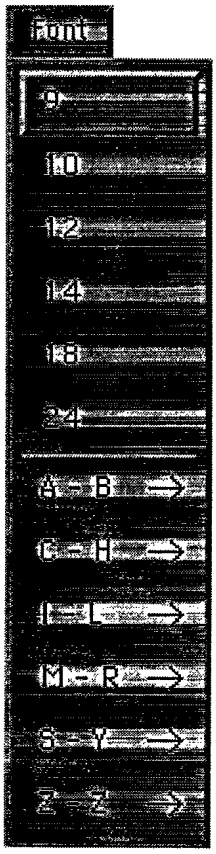
Show/Hide Ruler – a toggle command that shows or hides the workspace ruler.
Rulers... – displays the Workspace Attributes Window, which contains ruler characteristics.

Rotation – allows rotation of the currently selected object.

Zoom In – zooms in on the current design, using the last point clicked in the workspace as the centre for the zoom.

Zoom Out - zooms out on the current design, using the last point clicked in the workspace as the centre for the zoom.

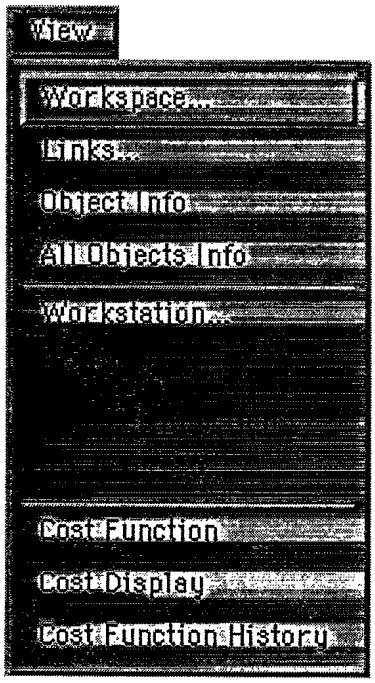
The Font Menu



<Font Size> - allows the choice of six font sizes.

<Font Type> –allows a choice of fonts recognised by the system. LOCATE detects the number of fonts in the user's system file and then organises them by sections of the alphabet in a way that allows all the available system fonts to be displayed.

The View Menu



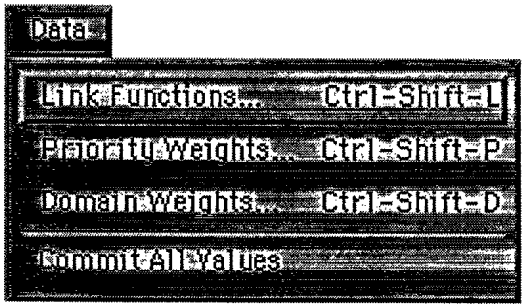
Workspace – displays the Workspace Attributes Window.
Links... - displays the Links Display Window, which allows you to display or hide links associated with the various communication domains.
Object Info – displays a window showing the attributes of the currently selected object.
All Objects Info – displays a window showing the attributes of all objects in the workspace that are of the same type as the currently selected object.

Workstation – displays the attributes window for the currently selected workstation.
Obstruction – displays the attributes window for the currently selected workstation obstruction.
Fixed Obstruction – displays the attributes window for the currently selected fixed obstruction.
Other Object – displays the attributes window for the currently selected object, when that object is not one of the above three objects.

Cost Function – displays the cost function window.
Cost Display – displays a colour cost display for the combined communication domains being analysed. Cost displays for the individual domains are accessible from the combined display.

Cost Function History – displays a list of all cost functions that have been run, both within and across the designs created or opened during a session. Any design for which a cost function has been computed may be recalled by double-clicking on its associated cost function value.

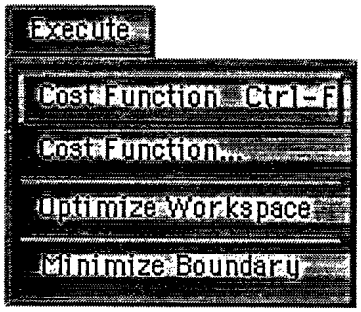
The Data Menu



Link Functions – opens the Link Functions portion of the Workstation Window for the currently selected workstation.
Priority Weights – opens the Priority Weights portion of the Workstation Window for the currently selected (pair of) workstation(s).
Domain Weights – opens the Domain Weights portion of the Workspace Attributes Window.

Commit All Values – commits all default priority weight values. If defaults remain after the user enters his or her priority weights, they must be committed before a cost function can be run.

The Execute Menu

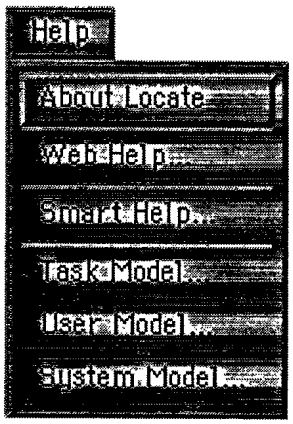


Cost Function – computes a cost function for the current design and displays it in the Cost Function Value Window at the bottom of LOCATE's palette.
Cost Function... – computes a cost function for the current design and displays it in a Cost Function Window. Contents of that Window allow for the display of detailed cost results for each communication domain being analysed and for the computation of cost functions to be done automatically by LOCATE. In automatic mode, a new cost function is computed whenever a change is made to the design.

Optimize Workspace – invokes a built-in optimiser that optimises the current design.

Minimize Boundary – minimises the boundary of a selected workstation or obstruction.

The Help Menu



About Locate – provides a brief introduction to LOCATE as well as access to more detailed LOCATE information and to its hypertext help system.

About You... – displays a window that allows you to identify yourself to LOCATE, instruct it to track your activities, provide you with feedback and remind you of this intelligent aiding option whenever LOCATE is opened.

Smart Help –allows you to ask about how some task can be done and provides feedback on the goals and actions necessary to make that happen. Information also can be provided on how the system will support you in that context.

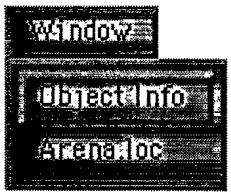
Web Help – provides access through LOCATE's built-in Web Browser to its hypertext help system, which is stored at a demonstration site on the World Wide Web.

Task Model – displays a window that contains LOCATE's beliefs about the task. The window is divided into three areas, containing: 1) the latest interface action taken by the user; 2) the user goal(s) implied by that action; and, 3) a goal and action history that contains all of the actions and implied goals that have occurred during a LOCATE session.

User Model – displays a window that contains LOCATE's beliefs about what the user knows. Those beliefs are derived from three sources: 1) what the user has told LOCATE directly; 2) inferences based on the help that LOCATE has provided to the user; and, 3) inferences based on LOCATE's tracking of user actions at the interface.

System Model – displays a window that contains LOCATE's beliefs about itself. Primarily that includes information about what whether LOCATE is able to help you whenever you request help and how it is able to do so. The beliefs in this Model are the ones most closely tied to LOCATE's help system, with entries generated largely as a consequence of your selecting the "Smart Help" command, described above, and asking LOCATE for help on how to perform some task.

Window Menu



This Menu displays all open LOCATE windows. If some windows are obscuring other windows, this Menu makes it possible to bring any one of the obscured windows to the front and make it the active Window. In the figure of the Menu, above, the choices are the Object Info window (see View Menu, above) and a Design Window entitled, "Arena.loc."

KEY WINDOWS IN LOCATE

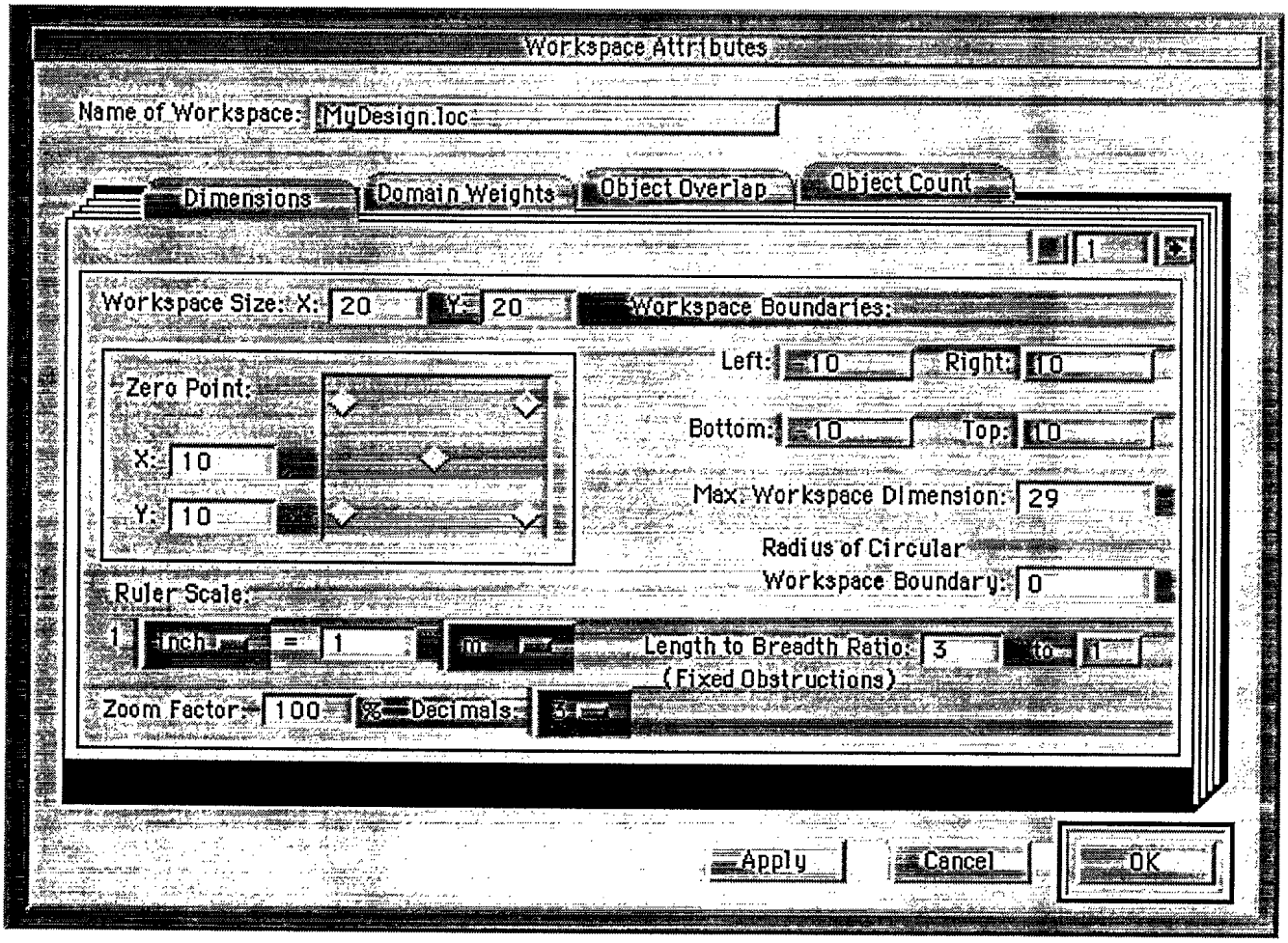
When using the LOCATE Workspace Layout Tool, numerous windows are displayed in a variety of contexts. Two of those windows are important enough to present in this Quick Start Manual.

The first is the **Workspace Attributes Window**, divided into four tabbed sections; the second is the **Workstation Window**, divided into three tabbed sections.

Workspace Attributes Window

The Workspace Attributes Window consists of the following four sections:

- Dimensions;
- Domain Weights;
- Object Overlap;
- Object Count.



Dimensions Tab. Proceeding down the left side of the **Dimensions** portion of the Workspace Attributes Window, it is possible to change the X and Y dimensions of the Workspace Size. A zero point is specified by clicking on one of the five radio buttons in the rectangle to the right of the words, "Zero Point" to locate it at the top or bottom left, top or bottom right, or in the centre of the workspace. Alternatively, a customised location can be entered by typing in the X and Y values in the edit text boxes below the words, "Zero Point."

Next, the ruler scale can be set by selecting "inches" or "centimetres" in the pop-up menu on the left and then setting the equivalent to "inches", "centimetres", "metres" or "feet" on the right and by specifying a corresponding ratio in the edit text box between the two pop-up menus.

The default zoom factor is set to 100% but may be changed by typing in a value in the edit text box associated with that factor.

Next to the Zoom is another pop-up menu for setting the number of decimal values to be displayed in all of LOCATE windows. The choices range from "0" to "6" decimals.

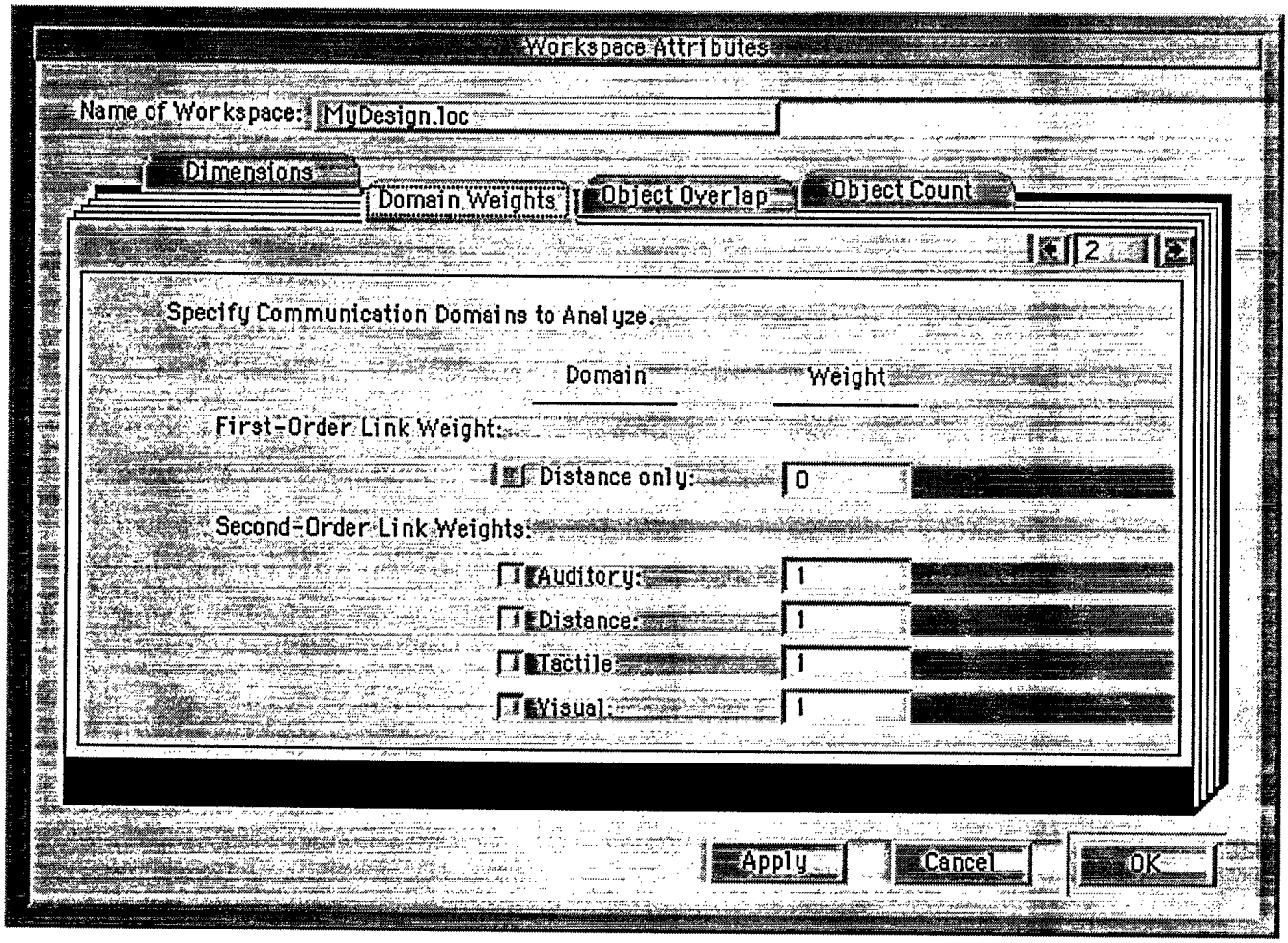
Proceeding down the right side of the Dimensions portion of the Workspace Attributes Window, the Workspace Boundaries are displayed in non-editable text boxes, which are derived from the workspace size and zero point on the left side of the window.

The Maximum Workspace Dimension is automatically calculated by LOCATE and is based on the largest distance across the workspace. You may override that value but any subsequent change in the dimensions of the "Workspace Size" will result in its being re-computed.

The "Radius of Circular Workspace Boundary' will have a value other than zero only if a portion of your workspace boundary is circular.

Finally, the "Length to Breadth Ratio {Fixed Obstruction}" identifies a value by which certain fixed obstructions are treated differently. If the ratio of any fixed obstruction is less than the value displayed (default = "3"), the boundary around such fixed obstructions will be a circle. If the ratio is greater than the value displayed, the boundary will have the appearance of a rounded rectangle.

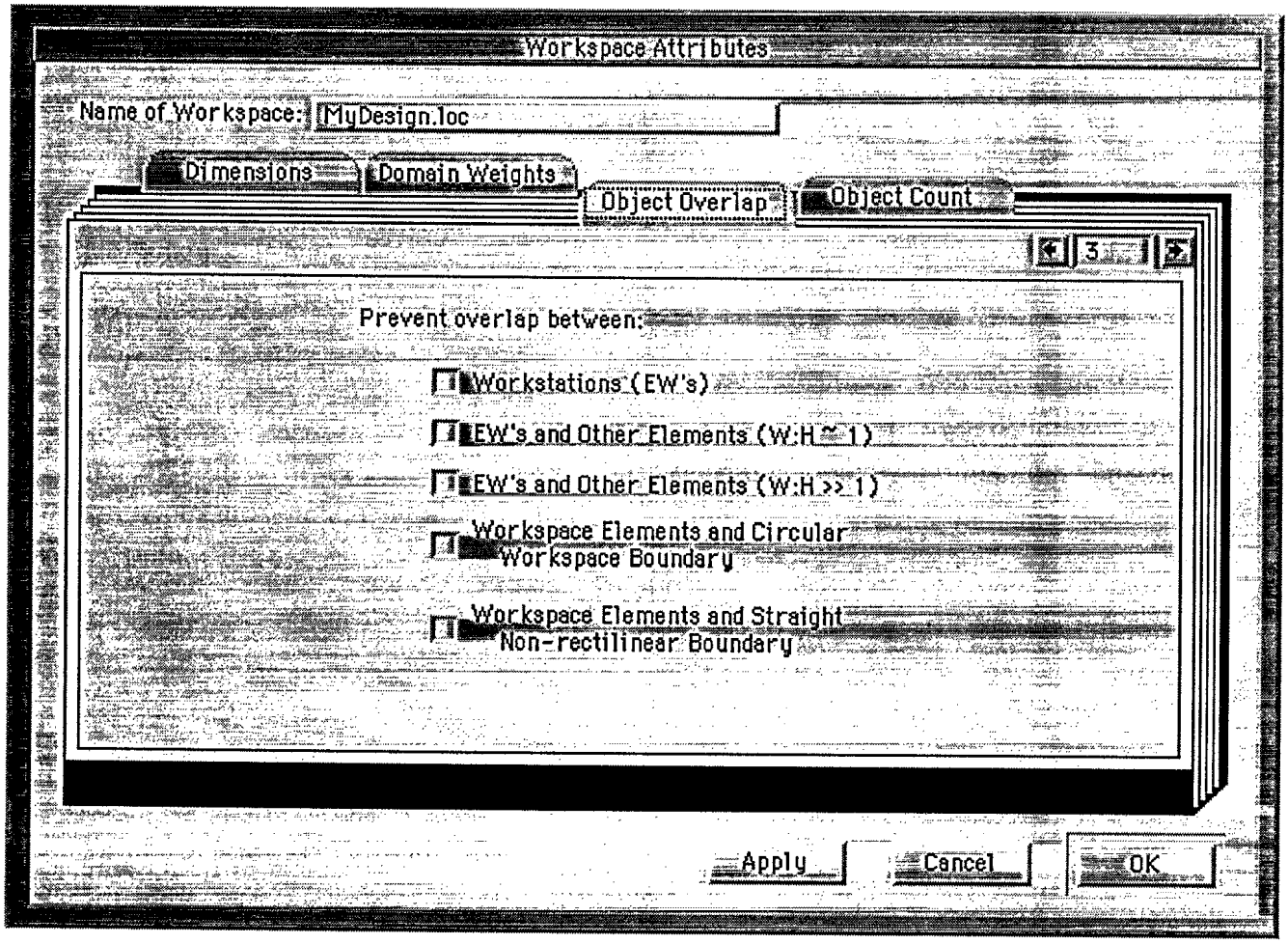
Domain Weights Tab. The **Domain Weights** portion of the Workspace Attributes Window, shown below, is where you identify which communication domains you wish to analyse. In most cases, some combination of the four communication domains that appear at the bottom of the window will be selected for analysis. They represent second-order link weights, which means that communication efficiency is assessed based on the relative positions of workstations in your design.



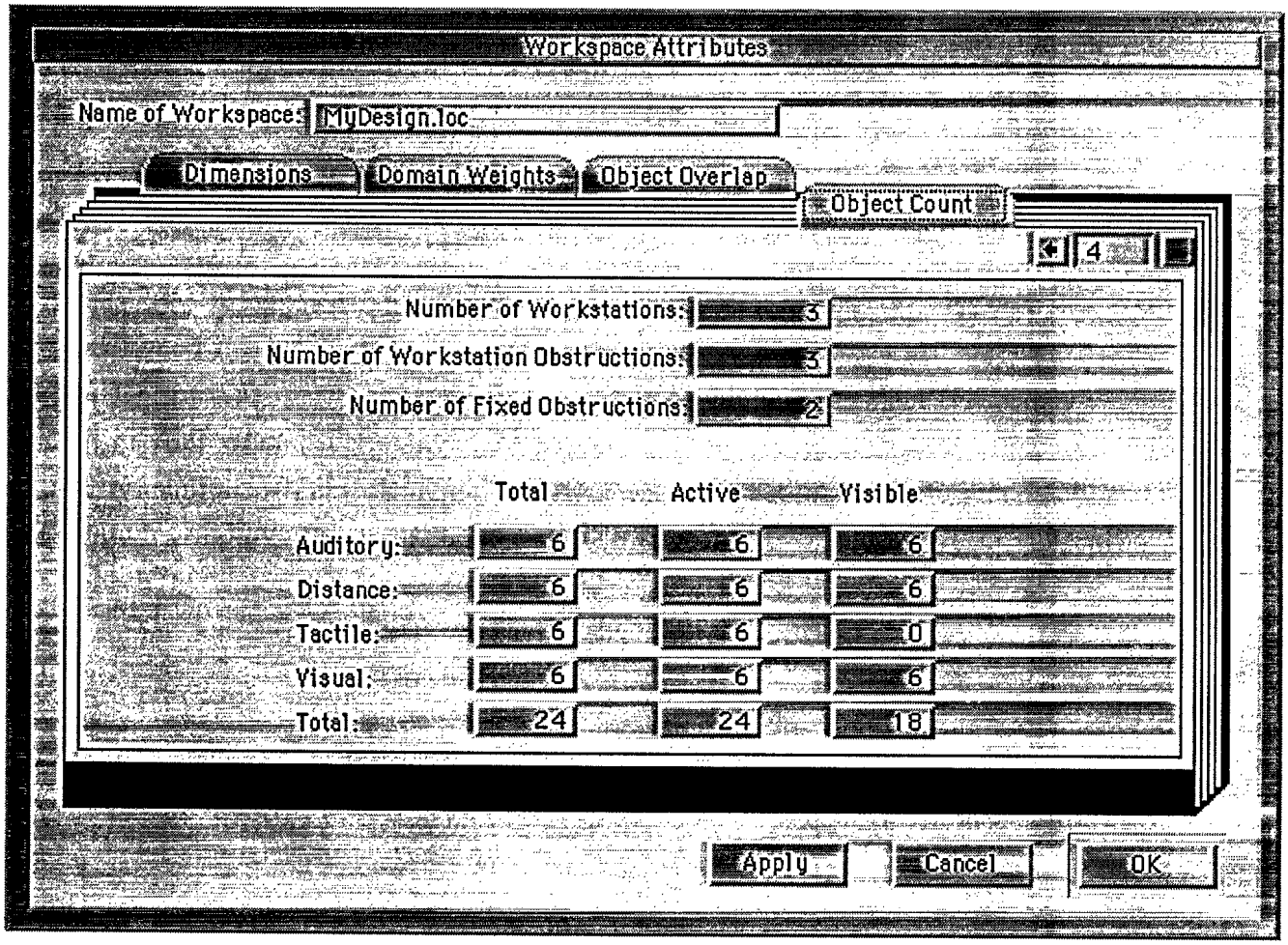
To the right of each communication domain name is the weight to be assigned. If the domain has not been selected, that value will be "0"; if selected, it will be "1." Those weights can take on other values, however, if more than one domain has been selected for analysis. The only limitation is that the sum total of all weights for the domains selected must be "1."

Finally, the "First-Order Link Weight" (Distance Domain only) is selected when system costs depend on absolute rather than relative positions of objects in the workspace, for example, when you wish to position elements judged to be particularly important at some location in the workspace, such as in the centre.

Object Overlap Tab. The third portion of the Workspace Attributes Window is the **Object Overlap** section. The default for all object types in the workspace portion is "to preventing overlap", but those settings are only taken into account during optimisation. Further, although hooks are available to implement those settings, they were not implemented for the example optimiser incorporated into LOCATE during this phase of the work.



Object Count Tab. The fourth and final portion of the Workspace Attributes Window is the **Object Count**. This is simply a summary of the number of object types that you have created in your workspace. The example figure below shows that the workspace contains three workstations, three (elemental) workstation obstructions, which appear only inside workstations, and two fixed obstructions, which appear only outside workstations and which are fixed during optimisation.



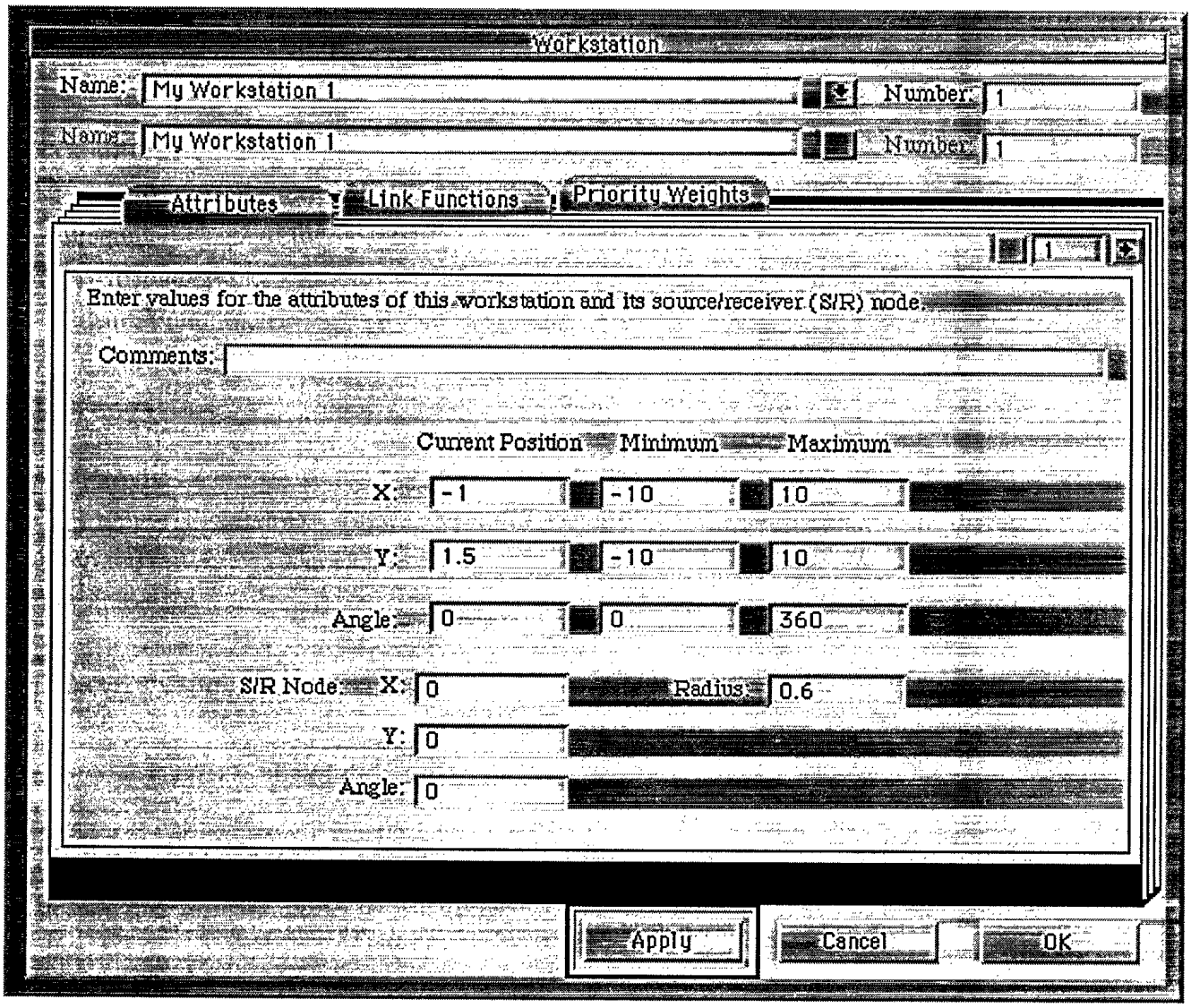
Regarding links, there are six of each type, all of which are active in the sense that they contribute to LOCATE's communication analysis, and all link types except tactile links are being displayed. Totals are given for all link types, for all active links and for all visible links.

Workstation Window

The Workstation Window is the second of the two important windows covered in this Manual and consists of three tabbed portions:

- Attributes;
- Link Functions;
- Priority Weights;

Attributes Tab. The **Attributes** portion displays the name, number and location of the workstation as well as its angle of rotation. Notice that the workstation name appears twice at the top of the display but the second instance is dimmed. That second instance is highlighted only when working in the Priority Weights section of this window (see below).



Minimum and maximum limits can be set for both location and angle. The defaults for those limits are the x and y dimensions of the workspace for position, and 0 to 360° for rotation.

The radius of the boundary that encloses the workstation appears in about the middle of the window at the right.

Each workstation has a point that represents the source and/or receiver of information in a workstation. That point is referred to as the "Source/Receiver" or "S/R" node, and has both location and angle within the local coordinates of the workstation. By default, the S/R node is placed at the centre of its workstation, and has local coordinates (0,0). Its rotation angle is set to "0" by default but it can be rotated independently of the workstation in which it resides.

Lastly, there is a place for entering comments about the workstation near the top of the Attributes portion of this Window.

The Link Functions Tab. The second portion of the Workstation Window is the **Link Functions** section shown in the figure on the next page.

Here is where functions are entered that describe the communication characteristics of a workstation. Functions and their arguments are entered for a workstation considered as a source and a receiver of information, for each domain being analysed, and for both distance and angular components of the workstation.

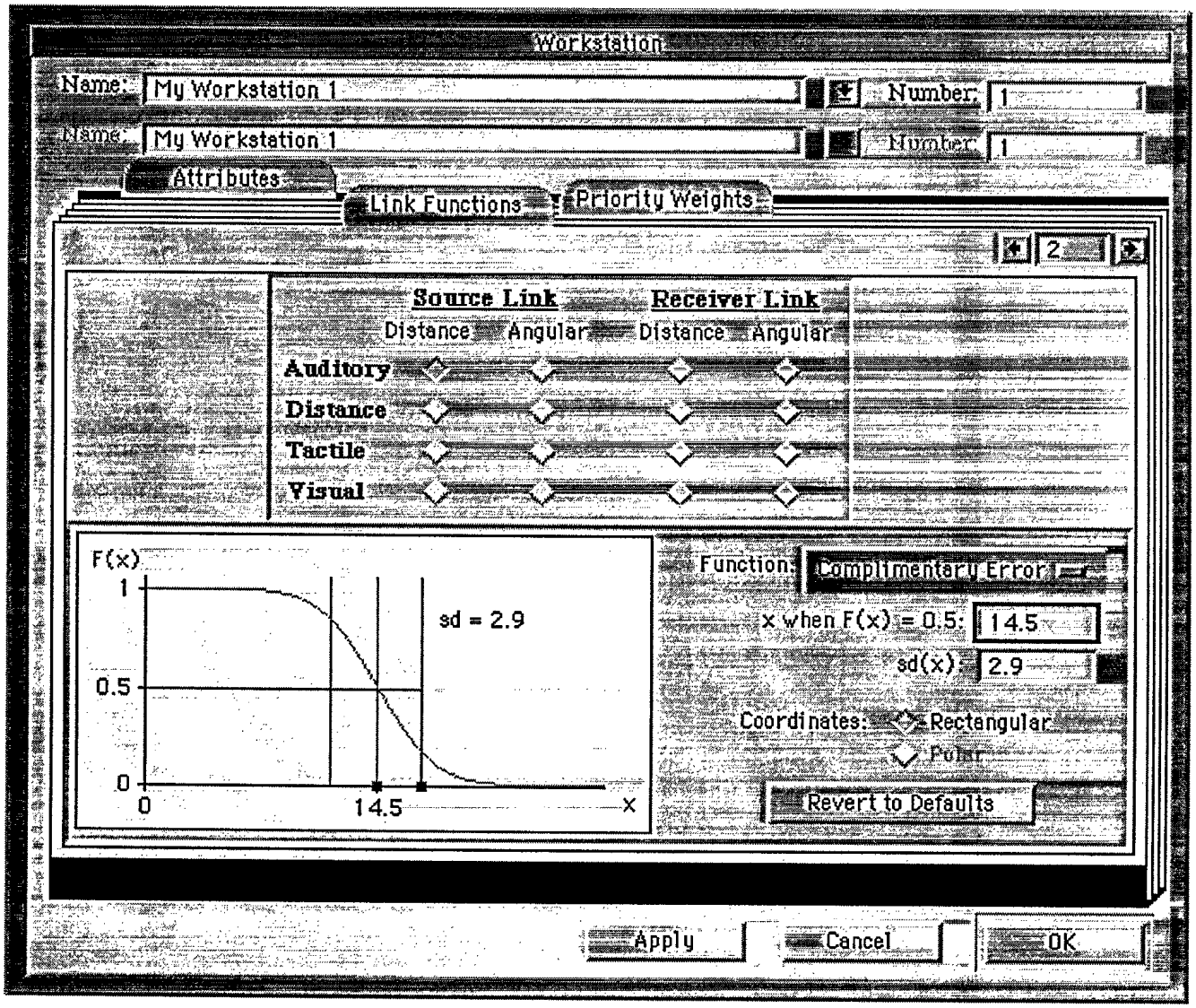
That means, if all four communication domains are to be analysed, sixteen functions and their arguments need to be entered for each workstation in the workspace.

Currently, seven functions are available to choose from in describing a distance component:

- butterworth;
- complimentary error;
- constant;
- exponential
- gaussian;
- inverse power;
- linear;

and three functions are available in describing the angular component:

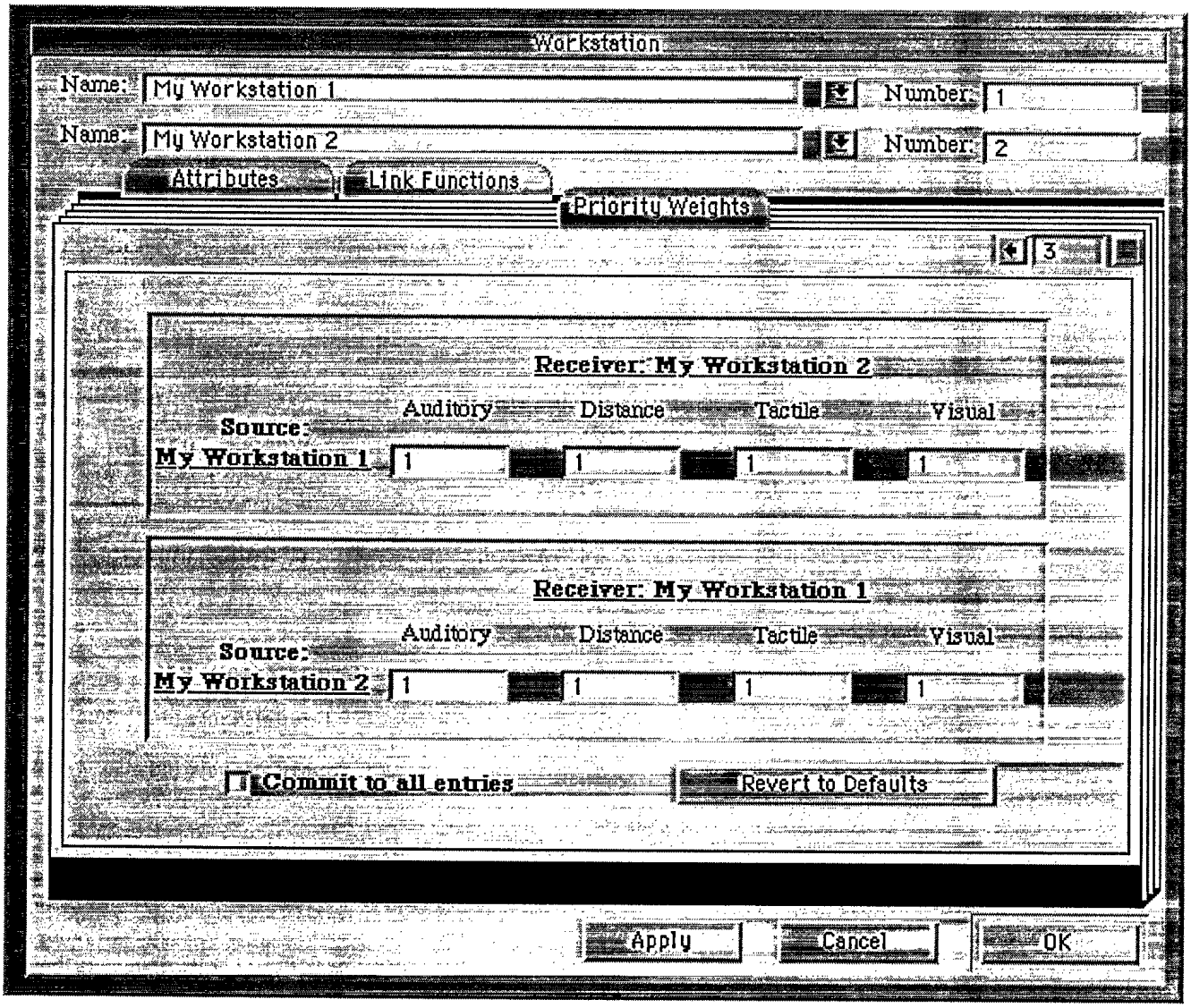
- butterworth;
- constant;
- gaussian;



Functions used to describe distance components are given rectangular coordinate representations only, but for those describing angular components, the representations may be in either rectangular or polar coordinates. Polar coordinate representations typically give a more intuitive "feel" for angular components and, for that reason, are the default representations for those components.

Further information describing these functions and how to determine which are appropriate to the particular set of workstations in your design is available through LOCATE's hypertext help system.

Priority Weights Tab. The third and final portion of the Workstation Window is the **Priority Weights** portion.



Priority weights are entered for each pair of workstations in a LOCATE workspace and define the importance of the various communication types for those workstations.

Priority weights range from –1 to +1 and are entered for each workstation in a pair, considered alternately, as the source and then as the receiver of information, relative to the other workstation in that pair. Positive values indicate that the communication type is more or less desirable, while negative values indicate that it is more or less undesirable.

In the section on Optimisation, that appears earlier in this Manual, there are options for specifying that certain workstations should be kept together and others kept apart. Selecting those options produces a condition that interacts with the priority weights, since both are indications of the need for communication between workstations.

The optimisation settings have a greater impact, however, than the weights, since the weights make a relatively small contribution to the overall cost function result, while the settings are checked and re-checked through a set of qualitative rules at each step in the optimisation process.

511943